Creación de aplicaciones de ADOBE® AIR®

Última modificación 1/5/2010

Creación de aplicaciones de Adobe® AIR®

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contenido

Capítulo 1: Presentación de Adobe AIR

Adobe® AIR® es un motor de ejecución válido para todos los sistemas operativos que le permite aprovechar sus habilidades de desarrollo web para crear e implementar aplicaciones enriquecidas de Internet (RIA) en el escritorio. Adobe® Flash® CS3 Professional, Adobe® Flash® CS4 Professional, Adobe® Flex™, Adobe® ActionScript® 3.0, HTML, JavaScript® y Ajax se pueden utilizar para crear aplicaciones de AIR.

Para obtener más información sobre el uso y una introducción a Adobe AIR, consulte el Centro de desarrollo de Adobe AIR http://www.adobe.com/devnet/air/.

AIR permite el trabajo en entornos conocidos para aprovechar las herramientas y los procesos con los que se encuentra más cómodo. Al admitir Flash, Flex, HTML, JavaScript y Ajax, es posible obtener la mejor experiencia posible que se adapte a sus necesidades.

Por ejemplo, se pueden desarrollar aplicaciones utilizando una de las tecnologías siguientes o combinando varias de ellas:

- Flash/Flex/ActionScript
- HTML/JavaScript/CSS/Ajax
- PDF (que puede aprovecharse con cualquier aplicación)

En consecuencia, las aplicaciones de AIR se pueden:

- Basar en Flash o Flex: aplicación cuyo contenido raíz es Flash/Flex (SWF).
- Basar en Flash o Flex con HTML o PDF. Aplicaciones cuyo contenido raíz es Flash/Flex (SWF) y que incluyen contenido HTML (HTML, JS, CSS) o PDF;
- Basar en HTML. Aplicación cuyo contenido raíz es HTML, JS, CSS.
- Basar en HTML con Flash/Flex o PDF. Aplicaciones cuyo contenido raíz es HTML y que incluyen contenido de Flash/Flex (SWF) o PDF.

Los usuarios interactúan con las aplicaciones de AIR de la misma forma en que interactúan con las aplicaciones de escritorio nativas. El motor de ejecución se instala una vez en el ordenador del usuario y después se instalan y ejecutan las aplicaciones de AIR como cualquier otra aplicación de escritorio.

El motor de ejecución ofrece una arquitectura y plataforma compatibles con distintos sistemas operativos para la implementación de aplicaciones. La compatibilidad y constancia del funcionamiento y las interacciones en distintos escritorios obvia la necesidad de realizar pruebas en distintos navegadores. En lugar de desarrollar programas para un sistema operativo determinado, el desarrollador centra sus esfuerzos en el motor de ejecución, lo cual ofrece las siguientes ventajas:

- Las aplicaciones desarrolladas para AIR se ejecutan en varios sistemas operativos distintos sin suponer trabajo adicional para el desarrollador. El motor de ejecución asegura una presentación e interacciones constantes y predecibles en todos los sistemas operativos compatibles con AIR.
- Las aplicaciones se pueden crear de forma más rápida permitiendo el aprovechamiento de tecnologías web y patrones de diseño existentes. Las aplicaciones basadas en web se pueden ampliar al escritorio sin tener que aprender las tecnologías de desarrollo en escritorio tradicionales o la complejidad del código nativo.
- El desarrollo de aplicaciones resulta más fácil que cuando se utilizan lenguajes de nivel inferior como C y C++. No hace falta gestionar las complejas API de nivel inferior que son específicas para cada sistema operativo.

Al desarrollar aplicaciones para AIR se puede aprovechar un juego enriquecido de arquitecturas e interfaces API:

- API específicas para AIR proporcionadas por el motor de ejecución y la arquitectura de AIR.
- API de ActionScript utilizadas en archivos SWF y la arquitectura de Flex (además de otras bibliotecas y arquitecturas basadas en ActionScript).
- HTML, CSS y JavaScript.
- La mayoría de las arquitecturas de Ajax.

AIR es toda una novedad en la forma de crear, implementar y experimentar las aplicaciones. Permite tener un mayor control creativo y extender al escritorio las aplicaciones basadas en Flash, Flex, HTML y Ajax sin necesidad de aprender las tradicionales tecnologías de desarrollo del escritorio.

Novedades de AIR 1.1

Adobe AIR 1.1 ha introducido las siguientes nuevas capacidades:

- La instalación y otros cuadros de diálogo del motor de ejecución se han traducido a los siguientes idiomas:
  - Portugués brasileño
  - Chino (tradicional y simplificado)
  - Francés
  - Alemán
  - Italiano
  - Japonés
  - Coreano
  - Ruso
  - Francés
  - Español
- Compatibilidad con la creación de aplicaciones internacionalizadas, incluyendo acciones de teclado para idiomas de doble byte. Consulte "Localización de aplicaciones de AIR" en la página 142.
  - Compatibilidad para localizar los atributos de nombre y descripción en el archivo descriptor de la aplicación.
  - Compatibilidad para localizar mensajes de error como, por ejemplo, SQLError.detailID y SQLError.detailArguments, en la base de datos SQLite.
  - Incorporación de la propiedad Capabilities.languages para obtener un conjunto de idiomas de IU preferidos como se establece mediante el sistema operativo.
  - Los menús predeterminados y las etiquetas de botón HTML como, por ejemplo, menús contextuales y la barra de menú de Mac, se han localizado a todos los idiomas compatibles.
- Compatibilidad para la migración de certificados de una aplicación con firma automática a otra que se encadena con una entidad emisora de certificados (CA).
- Compatibilidad con Microsoft Windows XP Tablet PC Edition y con ediciones de 64 bits de Windows Vista® Home Premium, Business, Ultimate o Enterprise.
- Incorporación de la API `File.spaceAvailable` para obtener la cantidad de espacio en disco disponible en un disco.

- Incorporación de la propiedad `NativeWindow.supportsTransparency` para establecer si una ventana se puede dibujar como transparente mediante el sistema operativo actual.

Para obtener más información sobre la versión de AIR 1.1, consulte las notas de versión de Adobe AIR 1.1 (http://www.adobe.com/go/learn_air_relnotes_es).

Novedades de AIR 1.5

Adobe AIR 1.5 presenta las siguientes nuevas funciones:

- Compatibilidad con las siguientes funciones de Flash Player 10.
  - Efectos y filtros personalizados
  - API de dibujo avanzada
  - Generación dinámica de sonido
  - Tipo de datos vectoriales
  - API mejorada de carga y descarga de archivos
  - Protocolo RTMFP (Real Time Media Flow Protocol) de flujo de medios en tiempo real
  - Efectos 3D
  - Compatibilidad para texto avanzado
  - Administración de color
  - Motor de texto
  - Transmisión de flujo dinámica
  - Códec de audio Speex

  Para obtener más información, consulte http://www.adobe.com/es/products/flashplayer/features/ para ver más datos sobre estas funciones.

  Para obtener información sobre el uso de las nuevas API ActionScript 3.0 de Flash Player 10, consulte la guía del desarrollador de Adobe ActionScript 3 (www.adobe.com/go/learn_flex_programmingAS3_es) y Adobe ActionScript 3.0 Reference for the Adobe Flash Platform (www.adobe.com/go/learn_flex3_aslr_es).
- Idiomas adicionales admitidos en el instalador de AIR 1.5 y otros cuadros de diálogo del motor de ejecución:
  - Checo
  - Neerlandés
  - Sueco
  - Turco
  - Polaco
- Cifrado de la base de datos.

  Los archivos de base de datos se pueden cifrar en AIR 1.5. Todo el contenido de base de datos, incluyendo los metadatos, se pueden cifrar para que la información no se pueda leer fuera de la aplicación de AIR que la cifró. Esta función permitirá que los desarrolladores cifren, descifren y vuelvan a cifrar archivos de base de datos. Consulte Almacenamiento local cifrado (para desarrolladores de ActionScript) o Almacenamiento local cifrado (para desarrolladores de HTML).

- La versión de WebKit utilizada por Adobe AIR se ha actualizado y ahora incluye compatibilidad para el intérprete de JavaScript, SquirrelFish.
- Nuevas API de validación de firmas XML que se pueden utilizar para ayudar a verificar la integridad y la identidad del firmante de los datos o información. Para obtener más información, consulte Validación de la firma XML en AIR (para desarrolladores de ActionScript), o bien, consulte Validación de la firma XML en AIR (para desarrolladores de HTML).

Para obtener más información sobre la versión de AIR 1.5, consulte las notas de versión de Adobe AIR 1.5 (http://www.adobe.com/go/learn_air_relnotes_es).

Novedades de AIR 1.5.1

Adobe AIR 1.5.1 presenta las siguientes nuevas funciones:

Plug-in de Flash Player más reciente
AIR 1.5.1 incluye una versión actualizada del plug-in de Flash Player (10.0.22) que se utiliza al mostrar archivos SWF dentro de HTML. Para obtener más información, consulte http://www.adobe.com/support/documentation/en/flashplayer/releasenotes.html.

Nuevas APIs

`InvokeEvent.reason`

Esta nueva propiedad del evento InvokeEvent indica si el usuario inició la aplicación mediante el usuario o automáticamente al iniciar la sesión. La clase InvokeEventReason (en el paquete flash.desktop) define los dos posibles valores de cadena para la propiedad InvokeEvent.reason. InvokeEventReason.LOGIN define el caso de inicio de sesión; InvokeEventReason.STANDARD define el caso estándar.

`Capabilities.cpuArchitecture`

Esta nueva propiedad devuelve la arquitectura del procesador del equipo, como cadena (por ejemplo, "PowerPC" o "x86").

Si desea aprovechar estas nuevas API de AIR 1.5.1, actualice el descriptor de la aplicación para que utilice el espacio de nombres 1.5.1:

```
xmlns="http://ns.adobe.com/air/application/1.5.1"
```

Si no necesita utilizar estas nuevas API, no es necesario actualizar el descriptor de la aplicación. La aplicación podrá ejecutarse con AIR 1.5.1 cuando el usuario actualice la versión del motor de ejecución instalado en el sistema.

Para obtener más información sobre la versión de AIR 1.5.1, consulte las notas de versión de Adobe AIR 1.5.1 (http://www.adobe.com/go/learn_air_relnotes_es).

Novedades de AIR 1.5.2

Adobe AIR 1.5.2 presenta las siguientes nuevas funciones:

Nuevas APIs
Capabilities.supports32BitProcesses y Capabilities.supports64BitProcesses

Estas propiedades indican si el sistema admite procesos de 64 ó 32 bits.

`LocalConnection.isPerUser`

Esta propiedad indica si los objetos LocalConnection están en el ámbito del usuario actual (true) o son globalmente accesibles para todos los usuarios del equipo (false). Esta propiedad sólo afecta al contenido que se ejecuta en Mac OS; en otras plataformas se omite este parámetro. Por ejemplo, las conexiones locales en Windows y Linux siempre son por usuario. En versiones anteriores, todos los objetos LocalConnection en Mac OS contaban con un ámbito global. Por motivos de seguridad, establezca siempre esta propiedad en true, a menos que necesite conservar la compatibilidad con versiones anteriores. En futuras versiones, es probable que esta propiedad tenga un valor predeterminado de true.

`System.disposeXML()`

Este método estático hace que un objeto XML de ActionScript esté inmediatamente disponible para la eliminación de datos innecesarios. Este método elimina las conexiones principales y secundarias entre todos los nodos para el objeto XML especificado. Este método adopta un parámetro: que el objeto XML esté disponible para la eliminación de datos innecesarios. Utilice este método para garantizar una eliminación de memoria eficaz asociada a objetos XML.

Actualización del archivo descriptor de la aplicación

Actualice el archivo descriptor de la aplicación al espacio de nombres 1.5.2 para acceder a las nuevas API y funcionalidad de AIR 1.5.2. Para actualizar el espacio de nombres, cambie el atributo xmlns a:

```
xmlns="http://ns.adobe.com/air/application/1.5.2"
```

Novedades en la versión beta de AIR 2

Para obtener datos detallados sobre la versión beta de AIR 2, consulte la información sobre AIR 2 Beta en Adobe Labs (en inglés).

Capítulo 2: Instalación de Adobe AIR

Adobe® AIR® permite ejecutar aplicaciones de AIR en el escritorio. El motor de ejecución se puede instalar de cualquiera de las formas siguientes:

- Mediante la instalación independiente del motor de ejecución (sin instalar además una aplicación de AIR).
- A través de la instalación de una aplicación de AIR por primera vez (aparecerá un mensaje sugiriendo que se instale el motor de ejecución).
- Con la instalación de un entorno de desarrollo de AIR como el kit de desarrollo de software de AIR, Adobe® Flash® Builder™ o el kit de desarrollo de software (SDK) de Adobe Flex® (que incluye las herramientas de desarrollo de la línea de comandos de AIR).

  El motor de ejecución sólo necesita instalarse una vez en cada ordenador.

Los requisitos del sistema para instalar AIR y ejecutar aplicaciones de AIR se describen en: Adobe AIR: Requisitos del sistema (http://www.adobe.com/es/products/air/systemreqs/).

Tanto el instalador del motor de ejecución como el instalador de la aplicación de AIR, crean archivos de registro cuando instalan, actualizan o eliminan aplicaciones de AIR o el propio motor de ejecución de AIR. Puede consultar estos registros para ayudar a determinar la causa de cualquier problema de instalación. Consulte “Registros de instalación” en la página 96.

Instalación de Adobe AIR

Siga estas instrucciones para descargar e instalar las versiones de AIR para Windows, Mac OS X y Linux.

Para actualizar el motor de ejecución, el usuario debe tener privilegios administrativos para el equipo.

Instalación del motor de ejecución en un ordenador con Windows

1. Descargue el archivo de instalación del motor de ejecución.
2. Haga doble clic en el archivo de instalación del motor de ejecución.
3. Siga las indicaciones que aparecen en la ventana de instalación para llevar a cabo la instalación.

Instalación del motor de ejecución en un ordenador con Mac

1. Descargue el archivo de instalación del motor de ejecución.
2. Haga doble clic en el archivo de instalación del motor de ejecución.
3. Siga las indicaciones que aparecen en la ventana de instalación para llevar a cabo la instalación.
4. Si el instalador presenta una ventana de autenticación, escriba el nombre de usuario y la contraseña que utiliza para Mac OS.

Instalación del motor de ejecución en un ordenador con Linux

1. Descargue el archivo de instalación del motor de ejecución.
2. Establezca los permisos de archivo para que se pueda ejecutar la aplicación de instalación:

Desde una línea de comandos, puede definir los permisos de archivo con el comando `chmod +x installer.bin`. Algunas versiones de Linux permiten establecer permisos de archivo en el cuadro de diálogo de propiedades que se abre mediante un menú contextual.

3 Ejecute el instalador desde la línea de comandos o haciendo doble clic en el archivo de instalación.

4 Siga las indicaciones que aparecen en la ventana de instalación para llevar a cabo la instalación.

AIR se instala como paquete rpm o dpkg, con los nombres: `adobeairv.n` y `adobecerts`. La instalación requiere la ejecución de un servidor X. AIR registra el tipo mime: `application/vnd.adobe.air-application-installer-package+zip`.

Desinstalación de Adobe AIR

Una vez instalado el motor de ejecución, se puede desinstalar siguiendo los procedimientos que se explican a continuación.

Desinstalación del motor de ejecución en un equipo con Windows

1 En el menú Inicio de Windows, seleccione Configuración > Panel de control.

2 Seleccione la opción Agregar o quitar programas.

3 Seleccione "Adobe AIR" para desinstalar el motor de ejecución.

4 Haga clic en el botón Cambiar o quitar.

Desinstalación del motor de ejecución en un equipo con Mac

- Haga doble clic en el archivo de desinstalación de Adobe AIR, que se encuentra en la carpeta /Aplicaciones/Utilidades.

Desinstalación del motor de ejecución en un ordenador con Linux

Realice uno de los siguientes pasos:

- Seleccione el comando"Desinstalador de Adobe AIR" en el menú Aplicaciones.
- Ejecute el instalador de AIR con la opción `-uninstall`.
- Elimine los paquetes de AIR (`adobeairv.n` y `adobecerts`) con el administrador de paquetes.

Instalación y ejecución de las aplicaciones de AIR de muestra

Hay algunas aplicaciones de muestra a disposición para demostrar las funciones de AIR. Para tener acceso a las mismas e instalarlas, siga estas instrucciones:

1 Descargue y ejecute las aplicaciones de AIR de muestra. Están a disposición tanto las aplicaciones compiladas como el código fuente.

2 Para descargar y ejecutar una aplicación de muestra, haga clic en el botón Instalar ahora de la aplicación. Un mensaje indica instalar y ejecutar la aplicación.

3 Si opta por descargar aplicaciones de muestra y ejecutarlas más adelante, seleccione los vínculos de descarga. Las aplicaciones de AIR pueden ejecutarse en cualquier momento de la siguiente manera:

- En Windows, haga doble clic en el icono de la aplicación que se encuentra en el escritorio o seleccione la aplicación en el menú Inicio.
- En Mac OS, haga doble clic en el icono de la aplicación, que se instala por omisión en la carpeta Aplicaciones de su directorio de usuario (por ejemplo, en Macintosh HD/Usuarios/UsuarioFicticio/Aplicaciones/).
- En Linux, haga doble clic en el icono de la aplicación que se encuentra en el escritorio o seleccione la aplicación en el menú de aplicaciones. Las aplicaciones de AIR se instalan en su propia carpeta en el directorio `/opt`.

Nota: *revise las notas de versión de AIR por si hubiera alguna actualización de estas instrucciones. Puede encontrarlas en: http://www.adobe.com/go/learn_air_relnotes_es.*

Actualizaciones de Adobe AIR

De forma periódica, Adobe actualiza Adobe AIR con nuevas funciones o soluciones para problemas menores. La función de actualización y notificación automática permite que Adobe avise automáticamente a los usuarios del momento en que está disponible una versión actualizada de Adobe AIR.

Las actualizaciones a Adobe AIR garantizan que la aplicación funcione adecuadamente y pueden incluir cambios importantes para la seguridad. Adobe recomienda que los usuarios actualicen a la versión más reciente de Adobe AIR si existe una nueva versión disponible, especialmente cuando se implica la actualización de seguridad.

De forma predeterminada, cuando se inicia una aplicación de AIR, el motor de ejecución comprueba si una actualización está disponible. Esta comprobación se lleva a cabo si han pasado más de dos semanas desde la última búsqueda de actualizaciones. Si alguna actualización está disponible, AIR la descarga en segundo plano.

Los usuarios pueden desactivar la capacidad de actualización automática, utilizando la aplicación SettingsManager de AIR. La aplicación SettingsManager de AIR está disponible para descarga en http://airdownload.adobe.com/air/applications/SettingsManager/SettingsManager.air.

El proceso de instalación normal de Adobe AIR incluye la conexión a http://airinstall.adobe.com para enviar la información básica sobre el entorno de instalación como, por ejemplo, idioma y versión del sistema operativo. Esta información sólo se transmite una vez por instalación y permite que Adobe pueda confirmar que la instalación se ha realizado correctamente. No se transmitirá ni se recopilará información de identificación personal.

Capítulo 3: Funcionalidad específica de Adobe AIR

Adobe® AIR® incluye funcionalidad que no está disponible para el contenido SWF que se ejecuta en Adobe® Flash® Player.

Desarrolladores de HTML de AIR
Si está creando aplicaciones de AIR basadas en HTML, consulte Adobe AIR API Reference for HTML Developers para obtener la lista completa de las API disponibles en JavaScript mediante el archivo AIRAliases.js.

Clases de ActionScript 3.0 específicas de AIR

Las siguientes clases del motor de ejecución son específicas de Adobe AIR. No están disponibles para el contenido SWF que se ejecute en el navegador:

Clase	Paquete
AAAARecord	flash.net.dns
ApplicationUpdater	air.update
ApplicationUpdaterUI	air.update
ARecord	flash.net.dns
BrowserInvokeEvent	flash.events
CertificateStatus	flash.security
CompressionAlgorithm	flash.utils
DatagramSocket	flash.net
DatagramSocketDataEvent	flash.events
DNSResolver	flash.net.dns
DNSResolverEvent	flash.events
DockIcon	flash.desktop
DownloadErrorEvent	air.update.events
DRMAuthenticateEvent	flash.events
DRMManagerError	flash.errors
EncryptedLocalStore	flash.data
File	flash.filesystem
FileListEvent	flash.events
FileMode	flash.filesystem

Clase	Paquete
FileStream	flash.filesystem
FocusDirection	flash.display
Geolocation	flash.sensors
GeolocationEvent	flash.events
HTMLHistoryItem	flash.html
HTMLHost	flash.html
HTMLLoader	flash.html
HTMLPDFCapability	flash.html
HTMLUncaughtScriptExceptionEvent	flash.events
HTMLWindowCreateOptions	flash.html
Icon	flash.desktop
InteractiveIcon	flash.desktop
InterfaceAddress	flash.net
InvokeEvent	flash.events
InvokeEventReason	flash.desktop
MXRecord	flash.net.dns
NativeApplication	flash.desktop
NativeDragActions	flash.desktop
NativeDragEvent	flash.events
NativeDragManager	flash.desktop
NativeDragOptions	flash.desktop
NativeMenu	flash.display
NativeMenuItem	flash.display
NativeProcess	flash.desktop
NativeProcessExitEvent	flash.events
NativeProcessStartupInfo	flash.desktop
NativeWindow	flash.display
NativeWindowBoundsEvent	flash.events
NativeWindowDisplayState	flash.display
NativeWindowDisplayStateEvent	flash.events
NativeWindowInitOptions	flash.display
NativeWindowResize	flash.display
NativeWindowSystemChrome	flash.display
NativeWindowType	flash.display

Clase	Paquete
NetworkInfo	flash.net
NetworkInterface	flash.net
NotificationType	flash.desktop
OutputProgressEvent	flash.events
PaperSize	flash.printing
PrintMethod	flash.printing
PrintUIOptions	flash.printing
PTRRecord	flash.net.dns
ReferencesValidationSetting	flash.security
ResourceRecord	flash.net.dns
RevocationCheckSettings	flash.security
Screen	flash.display
ScreenMouseEvent	flash.events
SecureSocket	flash.net
SecureSocketMonitor	air.net
SecureSocketConnectEvent	flash.net
ServiceMonitor	air.net
SignatureStatus	flash.security
SignerTrustSettings	flash.security
SocketMonitor	air.net
SQLCollationType	flash.data
SQLColumnNameStyle	flash.data
SQLColumnSchema	flash.data
SQLConnection	flash.data
SQLError	flash.errors
SQLErrorEvent	flash.events
SQLErrorOperation	flash.errors
SQLEvent	flash.events
SQLIndexSchema	flash.data
SQLMode	flash.data
SQLResult	flash.data
SQLSchema	flash.data
SQLSchemaResult	flash.data
SQLStatement	flash.data

Clase	Paquete
SQLTableSchema	flash.data
SQLTransactionLockType	flash.data
SQLTriggerSchema	flash.data
SQLUpdateEvent	flash.events
SQLViewSchema	flash.data
SRVRecord	flash.net.dns
StageAspectRatio	flash.display
StageOrientation	flash.display
StageOrientationEvent	flash.events
StatusFileUpdateErrorEvent	air.update.events
StatusFileUpdateEvent	air.update.events
StatusUpdateErrorEvent	air.update.events
StatusUpdateEvent	air.update.events
StorageVolume	flash.filesystem
StorageVolumeChangeEvent	flash.events
StorageVolumeInfo	flash.filesystem
SystemTrayIcon	flash.desktop
UpdateEvent	air.update.events
Updater	flash.desktop
URLFilePromise	air.desktop
URLMonitor	air.net
URLRequestDefaults	flash.net
XMLSignatureValidator	flash.security

Asimismo, existen dos interfaces que son específicas de AIR:

- flash.desktop.IFilePromise
- flash.security.IURIDereferencer

Clases de Flash Player con funcionalidad específica de AIR

Las siguientes clases están disponibles para el contenido SWF que se ejecuta en el navegador, pero AIR ofrece propiedades o métodos adicionales:

Clase	Método o propiedad
Capabilities	`languages`
Clipboard	`supportsFilePromise`
ClipboardFormats	`BITMAP_FORMAT` FILE_LIST_FORMAT FILE_PROMISE_LIST_FORMAT URL_FORMAT
Event	`DISPLAYING` EXITING HTML_BOUNDS_CHANGE HTML_DOM_INITIALIZE HTML_RENDER LOCATION_CHANGE NETWORK_CHANGE USER_IDLE USER_PRESENT
FileReference	`uploadUnencoded()`

HTTPStatusEvent	`HTTP_RESPONSE_STATUS`
	`responseURL`
	`responseHeaders`

KeyboardEvent	`commandKey` `controlKey`
LoaderContext	`allowLoadBytesCodeExecution`
LoaderInfo	`parentSandboxBridge` `childSandboxBridge`
NetStream	`resetDRMVouchers()` `setDRMAuthenticationCredentials()`

Clase	Método o propiedad

PrintJob	`active`
	`copies`
	`firstPage`
	`isColor`
	`jobName`
	`lastPage`
	`maxPixelsPerInch`
	`paperArea`
	`printableArea`
	`printer`
	`printers`
	`supportsPageSetupDialog`
	maxPixelsPerInch

URLRequest	`followRedirects`
	`manageCookies`
	`shouldAuthenticate`
	`shouldCacheResponse`
	`userAgent`
	`userCache`
	`selectPaperSize()`
	`showPageSetupDialog()`
	`start2()`
	`terminate()`

PrintJobOptions	`resolution`
URLStream	`Evento httpResponseStatus`
Stage	`nativeWindow`
Security	`APPLICATION`

La mayoría de estos nuevos métodos y propiedades sólo están disponibles para contenido que se encuentra en el entorno limitado de seguridad de la aplicación de AIR. No obstante, los nuevos integrantes de las clases URLRequest también están disponibles para el contenido que se ejecuta en otros entornos limitados.

Los métodos `ByteArray.compress()` y `ByteArray.uncompress()` incluyen cada uno un nuevo parámetro, `algorithm`, que permite seleccionar entre la compresión deflate y zlib. Este parámetro está disponible únicamente en el contenido que se ejecuta en AIR.

Componentes de Flex específicos de AIR

Los siguientes componentes MX de Adobe® Flex™ están a disposición cuando se desarrolla contenido para Adobe AIR:

- FileEvent
- FileSystemComboBox
- FileSystemDataGrid
- FileSystemEnumerationMode
- FileSystemHistoryButton
- FileSystemList
- FileSystemSizeDisplayMode
- FileSystemTree
- FlexNativeMenu
- HTML
- Window
- WindowedApplication
- WindowedSystemManager

Asimismo, Flex 4 incluye los siguientes componentes spark de AIR:

- Window
- WindowedApplication

Para obtener más información sobre los componentes de Flex para AIR, consulte Using the Flex AIR components (Uso de componentes de Flex para AIR).

Capítulo 4: Herramientas de la Plataforma Adobe Flash para el desarrollo de AIR

Es posible desarrollar aplicaciones de AIR con las siguientes herramientas de desarrollo de la Plataforma Adobe Flash.

Para los desarrolladores de ActionScript 3.0 (Flash y Flex):

- Adobe Flash Professional CS3, CS4, CS5
- SDKs de Adobe Flex 3.x y 4.0
- Adobe Flex Builder 3.x
- Adobe Flash Builder 4.0

Para desarrolladores de HTML y Ajax:

- SDK de Adobe AIR
- Adobe Dreamweaver CS3, CS4, CS5

Configuración de Flash CS3 para Adobe AIR

La actualización de Adobe® AIR™ para Adobe® Flash® CS3 Professional aumenta el entorno de desarrollo de Flash con elementos que permiten al usuario crear aplicaciones de AIR con Flash. Permite crear, probar y depurar archivos de aplicaciones de AIR en Flash.

Adobe® Flash® CS4 Professional puede crear aplicaciones de AIR, ya que lo admite de forma nativa. Para obtener más información, consulte Publicación para Adobe AIR en Utilización de Flash.

Nota: *la actualización de Adobe AIR para Flash CS3 admite AIR 1.0 y 1.1 y Flash Player 9.x. Flash CS4 es necesario para desarrollar aplicaciones con AIR 1.5 y Flash Player 10.*

Requisitos del sistema para la Actualización de Adobe AIR para Flash CS3

Para poder utilizar Flash CS3 en el desarrollo y la ejecución de aplicaciones de AIR, debe tener el siguiente software instalado:

- Flash CS3 Professional

  Si no dispone de una copia de Flash CS3 Professional, puede adquirirla en el sitio web de Adobe: http://www.adobe.com/es/products/flash/

- Adobe AIR

  Para obtener información sobre la instalación de Adobe AIR, consulte “Instalación de Adobe AIR” en la página 6.

- Actualización de Adobe AIR para Flash CS3

Instalación de la actualización de Adobe AIR 1.0 para Flash CS 33

Antes de instalar la actualización de Adobe AIR para Flash CS3, salga de Flash y de todos los navegadores que tenga abiertos.

- Descargue la actualización de Adobe AIR 1.0 para Flash CS3 (http://www.adobe.com/support/flash/downloads.html#flashCS3)
- Una vez descargada la actualización, haga doble clic en el archivo para instalarlo.

Instalación de la actualización de Adobe AIR 1.1 para Flash CS 33

Para instalar esta actualización, en primer lugar debe instalar la actualización de Adobe AIR 1.0 original para Adobe Flash CS3 Professional. No es posible instalar esta actualización sin instalar la previa.

1. Localice la carpeta denominada "AIK":

   Windows: C:/Archivos de programa/Adobe/Adobe Flash CS3

   Mac OS X: Macintosh HD: Aplicaciones: Adobe Flash CS3

2. Elimine todos los archivos de la carpeta AIK. No elimine la propia carpeta.
3. Descargue AIRSDKIntegrationKit de:

   Windows: http://www.adobe.com/go/getaikwin

   Mac OS X: http://www.adobe.com/go/getaikmac

4. Descomprima el contenido de AIRSDKIntegrationKit y sitúelo dentro de la carpeta AIK.
5. Descargue el motor de ejecución de Adobe AIR 1.1, en http://get.adobe.com/es/air/.
6. Instale el motor de ejecución en su equipo.

Desinstalación de la actualización de Adobe AIR para Flash CS3

Para desinstalar la actualización de Adobe AIR para Flash CS3, siga estos pasos:

1. Elimine la siguiente carpeta:

   (Windows) Disco duro:\Archivos de programa\Adobe\Adobe Flash CS3\AIK

   (Mac) HD:/Aplicaciones/Adobe Flash CS3/AIK

2. Vaya a la siguiente ubicación:

   (Windows) Disco duro:\Archivos de programa\Adobe\Adobe Flash CS3\<idioma>\First Run\Commands\

   (Mac) HD:/Aplicaciones/Adobe Flash CS3/First Run/Commands

   y elimine las siguientes carpetas/archivos:

   - Carpeta AIR
   - AIR - Application and Installer Settings.jsfl
   - AIR - Create AIR File.jsfl

3. Elimine el siguiente archivo:

   (Windows) Disco duro:\Archivos de programa\Adobe\Adobe Flash CS3\<idioma>\Configuration\External Libraries\FLAir.dll

   (Mac) HD:/Aplicaciones/Adobe Flash CS3/Configuration/External Libraries/FLAir.bundle.

4 Elimine el siguiente archivo:

(Windows) Disco duro:\Archivos de programa\Adobe\Adobe Flash CS3\<idioma>\Configuration\Players\AdobeAIR1_0.xml

(Mac) HD:/Aplicaciones/Adobe Flash CS3/Configuration/Players/ AdobeAIR1_0.xml

5 Vaya a la siguiente ubicación:

(Windows) Disco duro:\Document and Settings\<nombre de usuario>\Local Settings\Application Data\Adobe\Flash CS3\<idioma>\Configuration\Commands\

(Mac) HD:/Usuarios/<nombre de usuario>/Librería/Application Support/Adobe/Flash CS3/<idioma>/Configuration/Commands/

y elimine las siguientes carpetas/archivos:

- Carpeta AIR
- AIR - Application and Installer Settings.jsfl
- AIR - Create AIR File.jsfl

Nota: *si no ve la ubicación especificada en Windows, active la opción "Mostrar archivos/carpetas ocultos" en Opciones de carpeta.*

Desarrollo de aplicaciones de AIR con Flex Builder 3

Adobe® Flex™ Builder™ 3 proporciona las herramientas para crear aplicaciones de Adobe® AIR®. Flex Builder incluye los componentes de Flex para aplicaciones de Adobe AIR. El flujo de trabajo para el desarrollo de aplicaciones de AIR en Flex Builder es similar al que existe para el desarrollo de la mayoría de las aplicaciones basadas en Flex.

Creación de proyectos de AIR con Flex Builder

Si aún no la ha hecho, instale AIR y Flex Builder 3.

- Abra Flex Builder 3.
- Seleccione File (Archivo) > New (Nuevo)> Flex Project (Proyecto de Flex).
- Indique el nombre del proyecto.
- En Flex, las aplicaciones de AIR se consideran un tipo de aplicación. Se dispone de dos opciones. Una opción se destina a la creación de una aplicación basada en Flex que se ejecute en la web en Adobe® Flash® Player. La otra opción es la creación de una aplicación de AIR que se ejecute en el escritorio en Adobe AIR. Seleccione la aplicación de escritorio como tipo de aplicación.
- Seleccione la tecnología de servidor (si existe) que desee utilizar con la aplicación de AIR. Si no se va a utilizar ninguna tecnología de servidor, seleccione Ninguna y, a continuación, haga clic en Siguiente.
- Seleccione la carpeta en la que desee ubicar la aplicación. La carpeta predeterminada es bin. Haga clic en Siguiente.
- Modifique las rutas de biblioteca y origen según sea necesario y, a continuación, haga clic en Finalizar para crear el proyecto de AIR.

Depuración de aplicaciones de AIR con Flex Builder

Flex Builder proporciona una compatibilidad de depuración completa para las aplicaciones de AIR. Para obtener más información sobre las capacidades de depuración de Flex Builder, consulte la ayuda de esta aplicación.

1 Abra un archivo de origen para la aplicación (por ejemplo, un archivo MXML) en Flex Builder.

2 Haga clic en el botón Debug (Depurar) de la barra de herramientas principal.

También puede seleccionar Run > Debug.

La aplicación se inicia y se ejecuta en la aplicación de ADL (AIR Debugger Launcher). El depurador de Flex Builder detecta los puntos de corte o los errores en tiempo de ejecución y la aplicación se puede depurar como cualquier otra aplicación de Flex.

La aplicación también se puede depurar desde la línea de comandos, utilizando la herramienta de línea de comandos AIR Debug Launcher. Para obtener más información, consulte "Utilización de AIR Debug Launcher (ADL)" en la página 51.

Empaquetado de aplicaciones de AIR con Flex Builder

Cuando la aplicación se completa y está lista para su distribución (o se prueba con la ejecución en el escritorio), se empaqueta en un archivo de AIR. El empaquetado consta de los siguientes pasos:

- Seleccionar la aplicación de AIR que se desea publicar.
- De forma opcional, permitir a los usuarios ver el código fuente y posteriormente seleccionar qué archivos de la aplicación incluir.
- Firmar digitalmente la aplicación de AIR utilizando un certificado de firma de código comercial o creando y aplicando una firma automática.
- Opcionalmente crear un archivo de AIR intermedio, que puede firmarse con posterioridad.

Empaquetado de una aplicación de AIR

1 Abra el proyecto y asegúrese de que la aplicación no presente errores de compilación y se ejecute de forma prevista.

2 Seleccione Project (Proyecto) > Export Release Version (Exportar versión oficial).

3 Si tiene varios proyectos y aplicaciones abiertos en Flex Builder, debe seleccionar el proyecto específico de AIR que desee empaquetar.

4 Opcionalmente, seleccione Enable View Source (Habilitar vista de código fuente) si desea que los usuarios puedan ver el código fuente cuando ejecuten la aplicación. Puede seleccionar archivos individuales para excluir, seleccionando Choose Source Files (Seleccionar archivos de origen). De forma predeterminada, se seleccionan todos los archivos de origen. Para obtener más información sobre la publicación de archivos de origen en Flex Builder, consulte la ayuda de la aplicación.

5 De forma opcional, también puede cambiar el nombre del archivo de AIR que se genera. Cuando esté listo para continuar, haga clic en Next para firmar digitalmente la aplicación.

Firma digital de aplicaciones de AIR

Antes de continuar con la exportación de la versión oficial, decida si desea firmar la aplicación de AIR digitalmente. Dispone de varias opciones. La aplicación se puede firmar utilizando un certificado de firma de código comercial, se puede crear y utilizar un certificado de digital con firma automática, o bien, puede optar por empaquetar la aplicación ahora y firmarla más adelante.

Los certificados digitales emitidos por las entidades de emisión de certificados como, por ejemplo, VeriSign, Thawte, GlobalSign y ChosenSecurity garantizan al usuario su identidad como editor y comprueban que el archivo de instalación no se haya modificado desde la firma. Los certificados digitales con firma automática cumplen la misma función, pero no proporcionan validación por parte de un tercero.

También tiene la opción de empaquetar la aplicación de AIR sin una firma digital, creando un archivo de AIR intermedio (.airi). Un archivo de AIR intermedio no es válido porque no se puede instalar. Se utiliza en cambio para la comprobación (por parte del desarrollador) y se puede iniciar utilizando la herramienta de la línea de comandos ADT de AIR. AIR proporciona esta capacidad, ya que en algunos entornos de desarrollo un desarrollador o equipo concreto administran la firma. Con esta práctica se garantiza un nivel adicional de seguridad en la administración de certificados digitales.

Para obtener más información sobre el proceso de firma de aplicaciones, consulte "Firma digital de archivos de AIR" en la página 96.

Firma digital de la aplicación de AIR

1. Las aplicaciones de AIR se pueden firmar digitalmente seleccionando un certificado digital existente o creando un nuevo certificado con firma automática. Seleccione la opción Export and Sign an AIR File with a Digital Certificate (Exportar y firmar un archivo de AIR con un certificado digital).
2. Si ya dispone de un certificado digital existente, haga clic en Examinar y selecciónelo.
3. Para crear un nuevo certificado digital con firma automática, seleccione Create (Crear).
4. Introduzca la información necesaria y haga clic en Aceptar.
5. Haga clic en Next (Siguiente) para seleccionar opcionalmente los archivos para excluir del archivo de AIR exportado. De forma predeterminada, se incluyen todos los archivos.
6. Haga clic en Finish (Finalizar) para generar el archivo de AIR.

Creación de un archivo intermedio de AIR

- Seleccione la opción para exportar un archivo intermedio de AIR que se exportará más adelante. Haga clic en Finish para generar el archivo intermedio.

Una vez generado un archivo de AIR intermedio, se puede firmar utilizando la herramienta de la línea de comandos ADT (consulte "Firma de un archivo intermedio de AIR con ADT" en la página 68).

Creación de un proyecto de biblioteca de AIR

Para crear una biblioteca de código de AIR para varios proyectos de AIR, cree un proyecto de biblioteca de AIR utilizando el asistente para proyectos de biblioteca de Flex estándar.

1. Seleccione File (Archivo) > New (Nuevo)> Flex Library Project (Proyecto de biblioteca de Flex).
2. Especifique un nombre de proyecto.
3. Seleccione la opción Add Adobe AIR Libraries (Añadir bibliotecas de Adobe AIR) y, a continuación, haga clic en Next (Siguiente).

   Nota: *la versión del SDK de Flex que seleccione debe admitir AIR. El SDK de Flex 2.0.1 no es compatible.*
4. Modifique la ruta de compilación según sea necesario y, a continuación, haga clic en Finish. Para obtener más información sobre la creación de proyectos de biblioteca, consulte el tema "About library projects" sobre los proyectos de biblioteca de la ayuda de Flex Builder.

Instalación de la extensión de AIR para Dreamweaver

La extensión de AIR para Dreamweaver ayuda a crear aplicaciones de Internet completas para el escritorio. Por ejemplo, puede que se disponga que un conjunto de páginas web que interactúen entre sí para mostrar datos XML. La extensión de Adobe AIR para Dreamweaver se puede utilizar para empaquetar este conjunto de páginas en una pequeña aplicación que puede instalarse en un equipo de usuario. Cuando el usuario ejecuta la aplicación en su escritorio, la aplicación carga y muestra el sitio web en su propia ventana, independiente de un navegador. El usuario puede desplazarse por el sitio web localmente en su equipo sin conexión a Internet.

Las páginas dinámicas, como Adobe® ColdFusion® y PHP no se ejecutarán en Adobe AIR. El motor de ejecución sólo funciona con HTML y JavaScript. No obstante, JavaScript se puede utilizar en las páginas para llamar a cualquier servicio web expuesto en Internet (incluyendo servicios generados por PHP o ColdFusion) con métodos Ajax, tales como XMLHTTPRequest o APIs específicas de Adobe AIR.

Requisitos del sistema

Para utilizar la extensión de Adobe AIR para Dreamweaver, es necesario instalar y configurar adecuadamente el software siguiente:

- Dreamweaver CS3 o Dreamweaver CS4
- Adobe® Extension Manager CS3
- Java JRE 1.4 o posterior (necesario para la creación del archivo de Adobe AIR). Java JRE está disponible en http://java.sun.com/.

  Los requisitos anteriores sólo se aplican a la creación y previsualización de aplicaciones de Adobe AIR en Dreamweaver. Para instalar y ejecutar una aplicación de Adobe AIR en el escritorio, Adobe AIR también debe instalarse en el equipo. Para descargar el motor de ejecución, consulte www.adobe.com/go/es/air.

Instalación de la extensión de Adobe AIR para Dreamweaver

1. Descargue la extensión de Adobe AIR para Dreamweaver en http://www.adobe.com/es/products/air/tools/ajax/.
2. Haga doble clic en el archivo de extensión .mxp en el Explorador de Windows (Windows) o en Finder (Macintosh).
3. Siga las instrucciones que aparecen en pantalla para instalar la extensión.
4. Cuando haya finalizado, reinicie Dreamweaver.

   Para obtener información sobre el uso de la extensión de Adobe AIR para Dreamweaver, consulte Uso de la extensión de AIR para Dreamweaver.

Instalación del SDK de AIR

El SDK de Adobe AIR contiene las siguientes herramientas de la línea de comandos que se utilizan para iniciar y empaquetar aplicaciones:

AIR Debug Launcher (ADL) Permite ejecutar aplicaciones de AIR sin tener que instalarlas primero. Consulte "Utilización de AIR Debug Launcher (ADL)" en la página 51.

AIR Development Tool (ADT) Empaqueta aplicaciones de AIR en paquetes de instalación distribuibles. Consulte "Empaquetado de archivos de instalación de AIR con AIR Developer Tool (ADT)" en la página 54.

Las herramientas de la línea de comandos de AIR requieren Java para su instalación en el equipo. Puede utilizar la máquina virtual Java desde JRE o JDK (versión 1.4 o posterior). Java JRE y Java JDK se encuentran disponibles en http://java.sun.com/.

Nota: *Java no se requiere para los usuarios finales que ejecuten aplicaciones de AIR.*

Descarga e instalación del SDK de AIR

El SDK de AIR se puede descargar e instalar utilizando las siguientes instrucciones:

Instalación del SDK de AIR en Windows

- Descargue el archivo de instalación del SDK de AIR.
- El SDK de AIR se distribuye como archivo estándar. Para instalar AIR, extraiga el contenido del SDK en una carpeta del equipo (por ejemplo: C:\Archivos de programa\Adobe\AIRSDK o C:\AIRSDK).
- Las herramientas ADL y ADT se incluyen en la carpeta bin del SDK de AIR; añada la ruta a esta carpeta a la variable de entorno PATH.

Instalación del SDK de AIR en Mac OS X

- Descargue el archivo de instalación del SDK de AIR.
- El SDK de AIR se distribuye como archivo estándar. Para instalar AIR, extraiga el contenido del SDK en una carpeta del equipo (por ejemplo: /Users/<userName>/Applications/AIRSDK).
- Las herramientas ADL y ADT se incluyen en la carpeta bin del SDK de AIR; añada la ruta a esta carpeta a la variable de entorno PATH.

Instalación del SDK de AIR en Linux

- El SDK está disponible en el formato tbz2.
- Para instalar el SDK, cree una carpeta en la que desee descomprimirlo y, a continuación, utilice el siguiente comando: tar -jxvf <path to AIR-SDK.tbz2>

Para obtener información sobre instrucciones para comenzar a utilizar las herramientas del SDK de AIR, consulte Creación de aplicaciones de AIR con las herramientas de la línea de comandos.

Componentes del SDK de AIR

En la siguiente tabla se describen las funciones de los archivos incluidos en el SDK de AIR:

Carpeta del SDK	Descripción de las herramientas/archivos
BIN	adl.exe: AIR Debug Launcher (ADL) permite ejecutar una aplicación de AIR sin empaquetarla e instalarla primero. Para obtener más información sobre el uso de esta herramienta, consulte "Utilización de AIR Debug Launcher (ADL)" en la página 51. adt.bat: AIR Developer Tool (ADT) empaqueta la aplicación como archivo de AIR para distribución. Para obtener información sobre el uso de esta herramienta, consulte "Empaquetado de archivos de instalación de AIR con AIR Developer Tool (ADT)" en la página 54.
FRAMEWORKS	AIRAliases.js: proporciona definiciones "alias" que permiten acceder a las clases del motor de ejecución de ActionScript. Para obtener más información sobre el uso de este archivo alias, consulte Utilización del archivo AIRAliases.js. servicemonitor.swf: proporciona aplicaciones de AIR con medios basados en eventos de respuesta a cambios en conectividad de red en un host especificado. Para obtener más información sobre esta arquitectura, consulte Cambios de conectividad de red. (para desarrolladores de ActionScript) o Cambios de conectividad de red. (para desarrolladores de HTML).
LIB	adt.jar: archivo ejecutable adt, que se llama mediante el archivo adt.bat. Descriptor.1.0.xsd: archivo de esquema de la aplicación.
RUNTIME	Motor de ejecución de AIR: ADL utiliza el motor de ejecución para iniciar las aplicaciones de AIR antes de que se empaqueten o se instalen.
SAMPLES	Esta carpeta contiene un archivo descriptor de la aplicación de ejemplo, un ejemplo de la función de instalación integrada (badge.swf) y los iconos de la aplicación de AIR predeterminados; consulte "Distribución, instalación y ejecución de aplicaciones de AIR" en la página 86.
SRC	Esta carpeta contiene los archivos de origen para el ejemplo de instalación integrada.
TEMPLATES	descriptor-template.xml: una plantilla del archivo descriptor de la aplicación que es necesaria para todas las aplicaciones de AIR. Para obtener una descripción detallada del archivo descriptor de la aplicación, consulte "Configuración de las propiedades de una aplicación de AIR" en la página 71.

Configuración del SDK de Flex

Para desarrollar aplicaciones de Adobe® AIR® con Adobe® Flex™, dispone de las siguientes opciones:

- Puede descargar e instalar Adobe® Flash® Builder™, que proporciona herramientas integradas para crear proyectos de Adobe AIR y comprobar, depurar y empaquetar aplicaciones de AIR. Consulte "Creación de la primera aplicación de AIR de Flex en Flash Builder o Flex Builder" en la página 26.
- Puede descargar el SDK de Adobe® Flex™ y desarrollar aplicaciones de AIR de Flex utilizando su editor de texto favorito y las herramientas de la línea de comandos.

Información sobre las herramientas de la línea de comandos de AIR en el SDK de Flex

Cada una de las herramientas de la línea de comandos que se utiliza para crear una aplicación de Adobe AIR llama a la herramienta correspondiente utilizada para crear aplicaciones de Flex:

- amxmlc llama a mxmlc para compilar clase de la aplicación.
- acompc llama a compc para compilar clases de componente y biblioteca.
- aasdoc llama a asdoc para generar archivos de documentación a partir de comentarios de código de origen.

La única diferencia entre las versiones de las utilidades de Flex y AIR radica en que las versiones de AIR cargan opciones de configuración desde el archivo air-config.xml en lugar del archivo flex-config.xml.

Las herramientas del SDK de Flex y sus opciones de la línea de comandos se describen en su totalidad en Building and Deploying Flex Applications (Creación e implementación de aplicaciones de Flex) en la biblioteca de documentación de Flex. Las herramientas del SDK de Flex se describen aquí en un nivel básico como ayuda en su introducción y para destacar las diferencias existentes entre la creación de aplicaciones de Flex y la creación de aplicaciones de AIR.

Para obtener más información, consulte Creación de aplicaciones de AIR con las herramientas de la lí-nea de comandos.

Instalación del SDK de Flex

La creación de aplicaciones de AIR con las herramientas de la lí-nea de comandos requiere que Java esté instalado en el equipo. La máquina virtual Java se puede utilizar desde JRE o JDK (versión 1.5 o posterior). El JRE y JDK de Java están disponibles en http://java.sun.com/.

Nota: *Java no se requiere para los usuarios finales que ejecuten aplicaciones de AIR.*

El SDK de Flex proporciona la API de AIR y las herramientas de la línea de comandos que se utilizan para empaquetar, compilar y depurar las aplicaciones de AIR.

1. Si aún no lo ha hecho, descargue el SDK de Flex en http://opensource.adobe.com/wiki/display/flexsdk/Downloads.
2. Ubique el contenido del SDK en una carpeta (por ejemplo, Flex SDK).
3. Las utilidades de la línea de comandos se ubican en la carpeta bin.

Para obtener más información, consulte "Creación de la primera aplicación de AIR con el SDK de Flex" en la página 31.

Configuración del compilador

Generalmente se especifican opciones de compilación tanto en la línea de comandos como con uno o varios archivos de configuración. El archivo de configuración del SDK de Flex global contiene valores predeterminados que se utilizan siempre que se ejecutan los compiladores. Este archivo se puede editar para adaptarse a su entorno de desarrollo. Existen dos archivos de configuración de Flex globales ubicados en el directorio frameworks de la instalación del SDK de Flex. El archivo air-config.xml se usa cuando se ejecuta el compilador amxmlc. Este archivo configura el compilador para AIR incluyendo las bibliotecas de AIR. El archivo flex-config.xml se utiliza cuando se ejecuta mxmlc.

Los valores de configuración predeterminados resultan adecuados para detectar el modo de funcionamiento de Flex y AIR, pero cuando lleve a cabo proyectos de envergadura, analice más detalladamente las opciones disponibles. Se pueden proporcionar valores específicos del proyecto para las opciones del compilador en un archivo de configuración local que tenga prioridad sobre los valores globales para un proyecto determinado. Para obtener una lista completa de las opciones de compilación y para la sintaxis de los archivos de configuración, consulte Flex SDK Configuration in Building y Deploying Flex Applications (Configuración del SDK de Flex en la creación e implementación de aplicaciones de Flex) en la biblioteca de documentación de Flex.

Nota: *no se utilizan opciones de compilación específicamente para aplicaciones de AIR, pero se debe hacer referencia a las bibliotecas de AIR al compilar una aplicación de AIR. A estas bibliotecas se les suele hacer referencia en un archivo de configuración de nivel de proyecto, en un archivo para una herramienta de creación como Ant o directamente en la línea de comandos.*

Configuración del depurador

AIR admite la depuración directamente, por lo que no es necesario depurar una versión del motor de ejecución (tal y como se haría con Adobe® Flash® Player). Para dirigir la depuración de la línea de comandos, se utiliza Flash Debugger y AIR Debug Launcher (ADL).

Flash Debugger se distribuye en el directorio del SDK de Flex. Las versiones nativas, como fdb.exe en Windows, se encuentran en el subdirectorio bin. La versión de Java está en el subdirectorio lib. AIR Debug Launcher, adl.exe, está en el directorio bin de la instalación del SDK de Flex. (No existen ninguna versión de Java independiente).

Nota: *no es posible iniciar una aplicación de AIR directamente con fdb, ya que fdb intenta iniciarla con Flash Player. Deje que la aplicación de AIR se conecte a una sesión de fdb en ejecución.*

Para obtener más información, consulte "Utilización de AIR Debug Launcher (ADL)" en la página 51.

Configuración del empaquetador de aplicaciones

AIR Developer Tool (ADT), que empaqueta la aplicación en un archivo de AIR instalable, es un programa en Java. Sólo es necesario configurar el entorno para que la utilidad se pueda ejecutar correctamente.

El SDK incluye un archivo de script en el directorio bin del SDK para ejecutar ADT como comando. ADT también se puede ejecutar como programa en Java, lo cual puede resultar apropiado al utilizar herramientas de creación, como Apache Ant.

Para obtener más información, consulte "Empaquetado de archivos de instalación de AIR con AIR Developer Tool (ADT)" en la página 54.

Capítulo 5: Creación de la primera aplicación de AIR de Flex en Flash Builder o Flex Builder

Para obtener unas indicaciones rápidas y prácticas sobre el funcionamiento de Adobe® AIR®, utilice estas instrucciones para crear y empaquetar una sencilla aplicación "Hello World" de AIR basada en archivos SWF utilizando Adobe® Flash® Builder.

Si aún no lo ha hecho, descargue e instale Flex Builder 3. Para obtener más información, consulte "Configuración del SDK de Flex" en la página 23.

Creación de un proyecto de AIR

Flash Builder incluye las herramientas necesarias para desarrollar y empaquetar aplicaciones de AIR.

La creación de aplicaciones de AIR en Flash Builder o Flex Builder comienza del mismo modo en que se crean otros proyectos de aplicaciones basadas en Flex: mediante la definición de un nuevo proyecto.

Nota: *estas instrucciones hacen referencia a Flash Builder. (También funcionan en Flex Builder 3.)*

1. Abra Flash Builder.
2. Seleccione File (Archivo) > New (Nuevo)> Flex Project (Proyecto de Flex).
3. Indique el nombre del proyecto como AIRHelloWorld.
4. En Flex, las aplicaciones de AIR se consideran un tipo de aplicación. Se dispone de dos opciones:
   - Una aplicación de Flex que se ejecuta en la web en Adobe® Flash® Player.
   - Una aplicación de AIR que se ejecuta en el escritorio en Adobe AIR.

   Seleccione la aplicación de escritorio como tipo de aplicación.
5. No se utilizará ninguna tecnología de servidor, por lo que seleccione None (Ninguno) y haga clic en Next (Siguiente).
6. Seleccione la carpeta en la que desee ubicar la aplicación compilada. La carpeta predeterminada es bin. Haga clic en Finish (Finalizar) para crear el proyecto.

Los proyectos de AIR constan inicialmente de dos archivos: el archivo principal MXML y el archivo XML de la aplicación (al que se hace referencia como archivo descriptor de la aplicación). El último archivo especifica los parámetros para identificar, instalar e iniciar aplicaciones de AIR. Habrá ocasiones en las que se deseará editar manualmente este archivo. Por ahora, se debe tener en cuenta su presencia.

Para obtener más información, consulte Desarrollo de aplicaciones de AIR con Flex Builder y Desarrollo de aplicaciones de AIR con Flash Builder.

Escritura del código de aplicaciones de AIR

Para escribir el código de la aplicación "Hello World", se edita el archivo MXML de la aplicación (AIRHelloWorld.mxml), que se abre en el editor. Si no se abre, utilice el navegador de proyectos (Project Navigator) para abrir el archivo.

Las aplicaciones de AIR de Flex se incluyen en la etiqueta MXML WindowedApplication. La etiqueta MXML WindowedApplication crea una sencilla ventana que incluye controles básicos como, por ejemplo, una barra de título y un botón de cierre.

1 Añada un atributo `title` al componente WindowedApplication y asígnele el valor `"Hello World"`:

```
?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                       xmlns:s="library://ns.adobe.com/flex/spark"
                       xmlns:mx="library://ns.adobe.com/flex/mx"
                       title="Hello World">
</s:WindowedApplication>
```

2 Añada un componente Label a la aplicación (sitúelo dentro de la etiqueta WindowedApplication). Establezca la propiedad `text` del componente Label en `"Hello AIR"` y defina restricciones de diseño para mantenerlo centrado, tal y como se muestra a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                       xmlns:s="library://ns.adobe.com/flex/spark"
                       xmlns:mx="library://ns.adobe.com/flex/mx"
                       title="Hello World">

    <s:Label text="Hello AIR" horizontalCenter="0" verticalCenter="0"/>
</s:WindowedApplication>
```

3 Añada el siguiente bloque de estilo inmediatamente después de la etiqueta inicial WindowedApplication y antes de la etiqueta del componente label introduzca:

```
<fx:Style>
    @namespace s "library://ns.adobe.com/flex/spark";
    s|WindowedApplication
    {

skinClass:ClassReference("spark.skins.spark.SparkChromeWindowedApplicationSkin");
        background-color:#999999;
        background-alpha:"0.7";
    }
</fx:Style>
```

Esta configuración de estilo se aplica a toda la aplicación y se procesa el fondo de la ventana con un gris ligeramente transparente.

El código de la aplicación presenta en este momento el siguiente aspecto:

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                       xmlns:s="library://ns.adobe.com/flex/spark"
                       xmlns:mx="library://ns.adobe.com/flex/mx"
                       title="Hello World">

    <fx:Style>
        @namespace s "library://ns.adobe.com/flex/spark";
        s|WindowedApplication
        {

skinClass:ClassReference("spark.skins.spark.SparkChromeWindowedApplicationSkin");
            background-color:#999999;
            background-alpha:"0.7";
        }
    </fx:Style>

    <s:Label text="Hello AIR" horizontalCenter="0" verticalCenter="0"/>
</s:WindowedApplication>
```

A continuación, se modificará parte de la configuración para permitir que la aplicación sea transparente:

1. En el panel Flex Navigator (Navegador de Flex), sitúe el archivo descriptor de la aplicación en el directorio de origen del proyecto. Si se ha asignado el nombre AIRHelloWorld al proyecto, este archivo se denomina AIRHelloWorld-app.xml.
2. Haga doble clic en el archivo descriptor de la aplicación para editarlo en Flash Builder.
3. En el código XML, sitúe las líneas de comentarios para las propiedades `systemChrome` y `transparent` (de la propiedad `initialWindow`). Elimine los comentarios. (Elimine los delimitadores de comentarios `"<!--"` y `"-->"`.)
4. Establezca el valor de texto de la propiedad `systemChrome` en `none`, tal y como se muestra a continuación:

   ```
   <systemChrome>none</systemChrome>
   ```

5. Establezca el valor de texto de la propiedad `transparent` en `true`, tal y como se indica a continuación:

   ```
   <transparent>true</transparent>
   ```

6. Guarde el archivo.

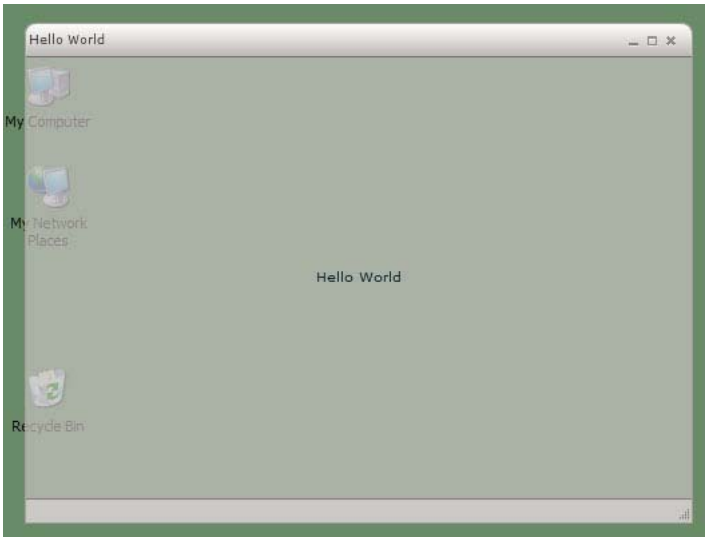
Prueba de la aplicación de AIR

Para probar el código de la aplicación que se ha escrito, ejecútelo en modo de depuración.

1 Haga clic en el botón Debug (Depurar) de la barra de herramientas principal de

También puede seleccionar el comando Run (Ejecutar) > Debug (Depurar) > AIRHelloWorld.

La aplicación de AIR resultante presentará el siguiente aspecto (el fondo verde es el escritorio):



2 Con el uso de las propiedades `horizontalCenter` y `verrticalCenter` del control Label, el texto se sitúa en el centro de la ventana. Mueva o cambie el tamaño de la ventana tal y como lo haría en cualquier otra aplicación de escritorio.

Nota: *si la aplicación no se compila, corrija la sintaxis o los errores ortográficos que se hayan podido introducir accidentalmente en el código. Los errores y advertencias se muestran en la vista Problems (Problemas) de Flash Builder.*

Empaquetado, firma y ejecución de una aplicación de AIR

Ahora ya se puede empaquetar la aplicación "Hello World" en un archivo de AIR para su distribución. Un archivo de AIR es un archivo de almacenamiento que contiene los archivos de la aplicación, que son todos los archivos incluidos en la carpeta bin del proyecto. En este sencillo ejemplo, estos archivos son los archivos SWF y XML de la aplicación. El paquete de AIR se distribuye a los usuarios, que posteriormente lo utilizan para instalar la aplicación. Un paso necesario en este proceso consiste en firmarlo digitalmente.

1 Compruebe que la aplicación no presenta errores de compilación y que se ejecuta correctamente.

2 Seleccione Project (Proyecto) > Export Release Version (Exportar versión oficial).

3 Si tiene varios proyectos y aplicaciones abiertos, debe seleccionar el proyecto específico de AIR que desee empaquetar. A continuación, haga clic en el botón Next (Siguiente).

4 Seleccione la opción Export and Sign an AIR File with a Digital Certificate (Exportar y firmar un archivo de AIR con un certificado digital).

5 Si ya dispone de un certificado digital existente, haga clic en Examinar y selecciónelo.

6 Si debe crear un nuevo certificado digital con firma automática, seleccione Crear.

7 Introduzca la información necesaria y haga clic en Aceptar.

8 Haga clic en Finalizar para generar el paquete de AIR denominado AIRHelloWorld.air.

Ahora puede ejecutar la aplicación desde Project Navigator (Navegador de proyectos) en Flash Builder o desde el sistema de archivos haciendo doble clic en el archivo de AIR.

Capítulo 6: Creación de la primera aplicación de AIR con el SDK de Flex

Para obtener unas indicaciones rápidas y prácticas sobre el funcionamiento de Adobe® AIR®, utilice estas instrucciones para crear una sencilla aplicación "Hello World" de AIR basada en SWF utilizando el SDK de Flex. Este tutorial muestra cómo compilar, probar y empaquetar una aplicación de AIR con las herramientas de la línea de comandos incluidas con el SDK.

Para comenzar, debe tener instalado el motor de ejecución y configurar Adobe® Flex™. En este tutorial se utiliza el compilador *AMXMLC*, *AIR Debug Launcher* (ADL) y *AIR Developer Tool* (ADT). Estos programas se pueden encontrar en el directorio `bin` del SDK de Flex (consulte "Configuración del SDK de Flex" en la página 23).

Creación del archivo descriptor de la aplicación de AIR

En esta sección se describe cómo crear el descriptor de la aplicación, que es un archivo XML con la siguiente estructura:

```
<application>
    <id>...</id>
    <version>...</version>
    <filename>…</filename>
    <initialWindow>
        <content>…</content>
        <visible>…</visible>
        <systemChrome>…</systemChrome>
        <transparent>…</transparent>
        <width>…</width>
        <height>…</height>
    </initialWindow>
</application>
```

1 Cree un archivo XML denominado `HelloWorld-app.xml` y guárdelo en el directorio del proyecto.

2 Añada el elemento `<application>`, incluyendo el atributo de espacio de nombres de AIR:

<application xmlns="http://ns.adobe.com/air/application/2.0"> El último segmento del espacio de nombres, "2.0", especifica la versión del motor de ejecución que requiere la aplicación.

3 Añada el elemento `<id>`:

<id>samples.flex.HelloWorld</id>El ID de la aplicación la identifica de forma exclusiva junto con el ID de editor (que AIR obtiene del certificado utilizado para firmar el paquete de la aplicación). La forma recomendada es una cadena de estilo DNS inversa delimitada por puntos como, por ejemplo, `"com.company.AppName"`. El ID de la aplicación se utiliza para la instalación, el acceso al directorio privado de almacenamiento del sistema de archivos de la aplicación, el acceso al almacenamiento cifrado privado y la comunicación entre aplicaciones.

4 Añada el elemento `<version>`:

<version>1.0</version> Ayuda a los usuarios a determinar qué versión de la aplicación se está instalando.

5 Agregue el elemento `<filename>`:

<filename>HelloWorld</filename> Nombre utilizado para el ejecutable de la aplicación, el directorio de instalación y otras referencias a la aplicación en el sistema operativo.

6 Añada el elemento `<initialWindow>` que contiene los siguientes elementos secundarios para especificar las propiedades de la ventana de la aplicación inicial:

<content>HelloWorld.swf</content> Identifica el archivo HTML raíz para que se cargue AIR.

<visible>true</visible> Hace visible a la ventana de forma inmediata.

<width>400</width> Establece la anchura de la ventana (en píxeles).

<height>200</height> Establece la altura de la ventana.

7 Guarde el archivo. El archivo descriptor de la aplicación completo debe presentar el siguiente aspecto:

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://ns.adobe.com/air/application/2.0">
    <id>samples.flex.HelloWorld</id>
    <version>0.1</version>
    <filename>HelloWorld</filename>
    <initialWindow>
        <content>HelloWorld.swf</content>
        <visible>true</visible>
        <systemChrome>none</systemChrome>
        <transparent>true</transparent>
        <width>400</width>
        <height>200</height>
    </initialWindow>
</application>
```

En este ejemplo sólo se establecen unas cuantas de las posibles propiedades de la aplicación. Para obtener el conjunto completo de las propiedades de la aplicación, que permiten especificar determinados aspectos, como el tamaño y el fondo cromático de la ventana, la transparencia, el directorio de instalación predeterminado, los tipos de archivo asociados y los iconos de la aplicación, consulte "Configuración de las propiedades de una aplicación de AIR" en la página 71

Escritura del código de la aplicación

Nota: *las aplicaciones de AIR basadas en SWF se pueden utilizar como clase principal definida con MXML o con Adobe® ActionScript® 3.0. En este ejemplo se utiliza un archivo MXML para definir su clase principal. El proceso para crear una aplicación de AIR con una clase ActionScript principal es similar. En lugar de compilar un archivo MXML en el archivo SWF, se compila el archivo de clase de ActionScript. Al utilizar ActionScript, la clase principal debe ampliar flash.display.Sprite.*

Al igual que sucede con todas las aplicaciones basadas en Flex, las aplicaciones de AIR creadas con la arquitectura de Flex contienen un archivo MXML principal. Sin embargo, las aplicaciones de AIR utilizan el componente *WindowedApplication* como elemento raíz en lugar del componente Application. El componente WindowedApplication proporciona propiedades, métodos y eventos para controlar la aplicación y su ventana inicial.

El siguiente procedimiento crea la aplicación Hello World:

1 Con el uso de un editor de texto, cree un archivo denominado `HelloWorld.mxml` y añada el siguiente código MXML:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="absolute" title="Hello World">
</mx:WindowedApplication>
```

2 A continuación, añada un componente Label a la aplicación (sitúelo dentro de la etiqueta WindowedApplication).

3 Establezca la propiedad `text` del componente Label en `"Hello AIR"`.

4 Defina las restricciones de diseño para mantenerlo siempre centrado.

En el siguiente ejemplo se muestra el código hasta el momento:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
    title="Hello World">
        <mx:Label text="Hello World" horizontalCenter="0" verticalCenter="0"/>
</mx:WindowedApplication>
```

5 Añada el siguiente bloque de estilo:

```
<mx:Style>
    WindowedApplication
    {
        background-color:"0x999999";
        background-alpha:"0.5";
    }
</mx:Style>
```

Estos estilos se aplican a toda la aplicación y el fondo de ventana se configura para que sea gris ligeramente transparente.

Todo el código de la aplicación presenta en este momento el siguiente aspecto:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute"
title="Hello World">
    <mx:Style>
        WindowedApplication
        {
            background-color:"0x999999";
            background-alpha:"0.5";
        }
    </mx:Style>
    <mx:Label text="Hello World" horizontalCenter="0" verticalCenter="0"/>
</mx:WindowedApplication>
```

Compilación de la aplicación

Antes de que se pueda ejecutar y depurar la aplicación, compile el código MXML en un archivo SWF utilizando el compilador amxmlc. El compilador amxmlc se encuentra en el directorio `bin` del SDK de Flex. Si lo desea, el entorno de ruta del equipo se puede configurar para que incluya el directorio bin del SDK de Flex. Al establecer la ruta, se facilita la ejecución de las utilidades en la línea de comandos.

1 Abra un shell de comandos o una terminal y desplácese a la carpeta del proyecto de la aplicación de AIR.

2 Indique el siguiente comando:

```
amxmlc HelloWorld.mxml
```

Con la ejecución de `amxmlc` se genera `HelloWorld.swf`, que contiene el código compilado de la aplicación.

Nota: *si la aplicación no se compila, corrija la sintaxis o los errores ortográficos. Los errores y los avisos se muestran en la ventana de la consola utilizada para ejecutar el compilador amxmlc.*

Para obtener más información, consulte "Compilación de archivos de origen MXML y ActionScript para AIR" en la página 48.

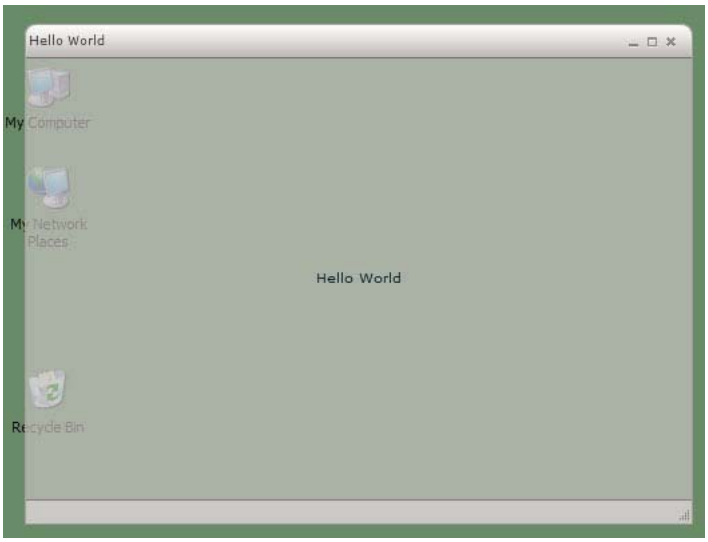
Prueba de la aplicación

Para ejecutar y probar la aplicación desde la línea de comandos, utilice AIR Debug Launcher (ADL) para iniciar la aplicación utilizando su archivo descriptor. (ADL se encuentra en el directorio bin del SDK de Flex.)

- Desde el símbolo del sistema, indique el siguiente comando:

  ```
  adl HelloWorld-app.xml
  ```

  La aplicación de AIR resultante presenta un aspecto similar al de esta ilustración (el fondo verde en el escritorio del usuario):



  Con el uso de las propiedades horizontalCenter y verrticalCenter del control Label, el texto se sitúa en el centro de la ventana. Mueva o cambie el tamaño de la ventana tal y como lo haría en cualquier otra aplicación de escritorio.

  Para obtener más información, consulte "Utilización de AIR Debug Launcher (ADL)" en la página 51.

Creación de un archivo de instalación de AIR

Cuando la aplicación se ejecute correctamente, puede emplear la utilidad ADT para empaquetar la aplicación en un archivo de instalación de AIR. Un archivo de instalación de AIR contiene todos los archivos de la aplicación, que se pueden distribuir a los usuarios. Se debe instalar Adobe AIR antes de instalar un archivo de AIR empaquetado.

Para garantizar la seguridad de la aplicación, todos los archivos de instalación de AIR se deben firmar digitalmente. Por motivos de desarrollo, se pueden generar certificados básicos con firma automática con ADT u otra herramienta de generación de certificados. También puede adquirir un certificado de firma de código comercial en una entidad emisora de certificados. Si los usuarios instalan un archivo de AIR con firma automática, el editor se muestra como "unknown" (desconocido) durante el proceso de instalación. Esto se debe a que el certificado con firma automática sólo garantiza que el archivo de AIR no se ha modificado desde su creación original. No existe ningún método para evitar que alguien firme automáticamente un archivo de AIR de enmascaramiento y lo presente como su aplicación. Para los archivos de AIR distribuidos públicamente, se recomienda el uso de un certificado comercial verificable. Para obtener información general sobre los problemas de seguridad en AIR, consulte Seguridad en AIR (para desarrolladores de ActionScript) o Seguridad en AIR (para desarrolladores de HTML).

Generación de un certificado con firma automática y un par de claves

❖ Desde el símbolo del sistema, indique el siguiente comando (el ejecutable de ADT se ubica en el directorio `bin` del SDK de Flex):

```
adt –certificate -cn SelfSigned 1024-RSA sampleCert.pfxsamplePassword
```

En este ejemplo se utiliza el número mínimo de atributos que se pueden establecer para un certificado. Se puede utilizar cualquier valor para los parámetros en *cursiva*. El tipo de clave debe ser *1024-RSA* o *2048-RSA* (consulte"Firma digital de archivos de AIR" en la página 96).

Creación de un archivo de instalación de AIR

❖ Desde el símbolo del sistema, introduzca el siguiente comando (en una sola línea):

```
adt -package -storetype pkcs12 -keystore sampleCert.pfx HelloWorld.air
HelloWorld-app.xml HelloWorld.swf
```

Se le solicitará la contraseña del archivo del almacén de claves.

El argumento HelloWorld.air es el archivo de AIR que genera ADT. HelloWorld-app.xml es el archivo descriptor de la aplicación. Los siguientes argumentos son los archivos utilizados por la aplicación. En este ejemplo sólo se utilizan tres archivos, pero se puede incluir cualquier número de archivos y directorios.

Una vez creado el paquete de AIR, se puede instalar y ejecutar la aplicación haciendo doble clic en el archivo del paquete. También se puede escribir el nombre del archivo de AIR como comando en una ventana de comandos o de shell.

Para obtener más información, consulte "Empaquetado de archivos de instalación de AIR con AIR Developer Tool (ADT)" en la página 54.

Capítulo 7: Creación de la primera aplicación de AIR con Flash Professional

A continuación se resume la demostración del funcionamiento de Adobe® AIR®. Siga las instrucciones de este tema para crear y empaquetar una sencilla aplicación "Hello World" de AIR con Adobe® Flash® Professional.

Creación de la aplicación Hello World en Flash

Crear una aplicación de Adobe AIR en Flash es muy similar a crear cualquier otro archivo FLA. Sin embargo, la creación de un archivo de Flash (Adobe AIR) comienza desde la pantalla de bienvenida. Se concluye estableciendo la configuración del instalador y de la aplicación e instalando la aplicación de AIR. El siguiente procedimiento le guiará en el proceso de creación de una sencilla aplicación Hello World con Flash Professional.

Para crear la aplicación Hello World

1. Inicie Flash.
2. En la pantalla de bienvenida, haga clic en Archivo de Flash (Adobe AIR) para crear un archivo FLA vacío con configuración de publicación de Adobe AIR.
3. Seleccione la herramienta Texto en el panel Herramientas y cree un campo de texto estático (valor predeterminado) en el centro del escenario. Dele una anchura suficiente para que pueda contener entre 15 y 20 caracteres.
4. Escriba el texto "Hello World" en el campo de texto.
5. Guarde el archivo y asígnele un nombre (por ejemplo, helloAIR).

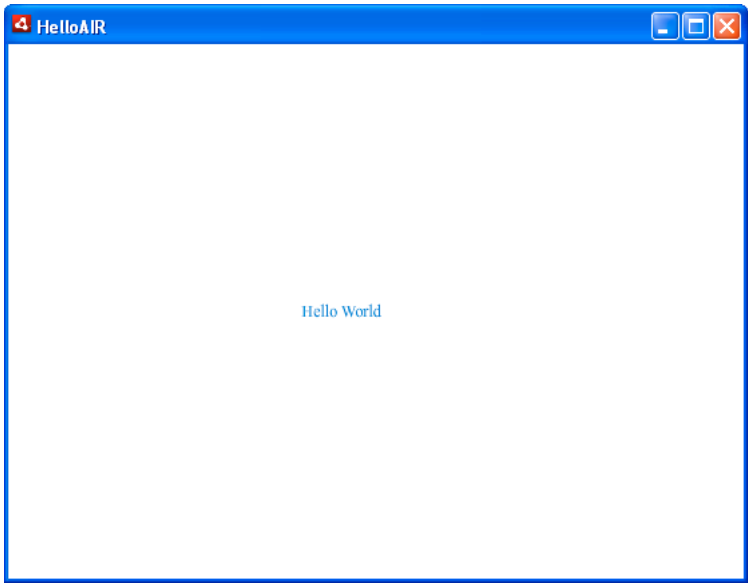
Prueba de la aplicación

1. Presione Ctrl + Intro o seleccione Control -> Probar película para probar la aplicación en Adobe AIR.
2. Para utilizar la función Depurar película, añada primero código ActionScript a la aplicación. Puede intentarlo rápidamente añadiendo una sentencia trace como ésta:

   ```
   trace("Running AIR application using Debug Movie");
   ```

3. Presione Ctrl + Mayús + Intro o seleccione Control -> Depurar película para ejecutar la aplicación con Depurar película.

La aplicación Hello World se asemeja a la de la ilustración:



Empaquetado de la aplicación

1 Seleccione Archivo > Publicar.

2 Firme el paquete de Adobe AIR con un certificado digital existente o cree un certificado con firma automática utilizando los siguientes pasos:

   a Haga clic en el botón Crear... para abrir el cuadro de diálogo Crear certificado digital con firma automática.

   b Rellene los campos Nombre del editor, Unidad de organización, Nombre de organización, Correo electrónico, País, Contraseña y Confirmar contraseña.

   c Especifique el tipo de certificado. La opción Tipo de certificado hace referencia al nivel de seguridad: 1024-RSA utiliza una clave de 1.024 bits (menos segunda) y 2048-RSA utiliza una clave de 2048 bits (más segura).

   d Guarde la información en un archivo de certificado en la opción Guardar como o haciendo clic en el botón Examinar... para acceder a la ubicación de la carpeta. (Por ejemplo, *C:/Temp/mycert.pfx*). Cuando haya terminado, haga clic en Aceptar.

   e Flash regresa al cuadro de diálogo Firma digital. La ruta y el nombre de archivo del certificado con firma automática creado aparece ahora en el cuadro de texto Certificado. Si no es así, introduzca la ruta y el nombre de archivo o haga clic en el botón Examinar para encontrarlo y seleccionarlo.

   f Indique la misma contraseña en el campo de texto Contraseña del cuadro de diálogo Firma digital que la que se asignó en el paso c. Para obtener más información sobre la firma de las aplicaciones de Adobe AIR, consulte "Firma digital de archivos de AIR" en la página 96.

3 Para crear el archivo aplicación y el instalador, haga clic en el botón Publicar archivo. (En Flash CS4, haga clic en el botón Aceptar.) Debe ejecutar los comandos Probar película o Depurar película para crear los archivos SWF y application.xml antes de crear el archivo de AIR.

4 Para instalar la aplicación, haga doble clic en el archivo de AIR (*application*.air) en la misma carpeta en la que guardó la aplicación.

5 Haga clic en el botón Instalar del cuadro de diálogo Instalación de la aplicación.

6 Revise los parámetros de Preferencias de instalación y Ubicación y asegúrese de que la casilla de verificación 'Iniciar aplicación tras la instalación' está seleccionada. A continuación, haga clic en Continuar.

7 Haga clic en Finalizar cuando aparezca el mensaje Instalación completada.

Capítulo 8: Creación de la primera aplicación de AIR basada en HTML con Dreamweaver

Para obtener unas indicaciones rápidas y prácticas sobre el funcionamiento de Adobe® AIR®, utilice estas instrucciones para crear y empaquetar una sencilla aplicación "Hello World" de AIR basada en HTML utilizando la extensión de Adobe® AIR® para Dreamweaver®.

Si aún no lo ha hecho, descargue e instale Adobe AIR en http://www.adobe.com/es/products/air/.

Para obtener instrucciones sobre la instalación de la extensión de Adobe AIR para Dreamweaver, consulte Instalación de la extensión de AIR para Dreamweaver.

Para obtener información general sobre la extensión, incluyendo los requisitos del sistema, consulte Extensión de AIR para Dreamweaver.

Preparación de los archivos de la aplicación

La aplicación de Adobe AIR debe contar con una página de inicio y todas sus páginas relacionadas definidas en un sitio de Dreamweaver:

1. Inicie Dreamweaver y asegúrese de disponer de un sitio definido.
2. Abra una nueva página HTML seleccionando Archivo > Nuevo, elija HTML en la columna Tipo de página, seleccione Ninguno en la columna Diseño y haga clic en Crear.
3. En la nueva página, escriba **Hello World!**

   Este ejemplo es muy sencillo, pero, si lo desea, puede aplicar el estilo que desee al texto, añadir más contenido a la página, vincular otras páginas a esta página de inicio, etc.
4. Guarde la página (Archivo > Guardar) como hello_world.html. Compruebe que el archivo se guarda en un sitio de Dreamweaver.

Para obtener más información sobre los sitios de Dreamweaver, consulte la ayuda de Dreamweaver.

Creación de una aplicación de Adobe AIR

1. Compruebe que la página hello_world.html esté abierta en la ventana de documento de Dreamweaver. (Consulte la sección anterior para obtener instrucciones sobre cómo crearla.)
2. Seleccione Sitio > Configuración de la aplicación de Air.

   La mayor parte de las opciones de configuración necesarias del cuadro de diálogo AIR - Configuración de aplicación e instalador se encuentran seleccionadas automáticamente. Sin embargo, se debe seleccionar el contenido inicial (o página de inicio) de la aplicación.
3. Haga clic en el botón Examinar situado junto a la opción Contenido inicial, desplácese a la página hello_world.html y selecciónela.

4 Junto a la opción Firma digital, haga clic en el botón Definir.

Con la firma digital se garantiza que el código de una aplicación no se ha alterado ni dañado desde su creación por parte del autor del software y es necesaria en todas las aplicaciones de Adobe AIR.

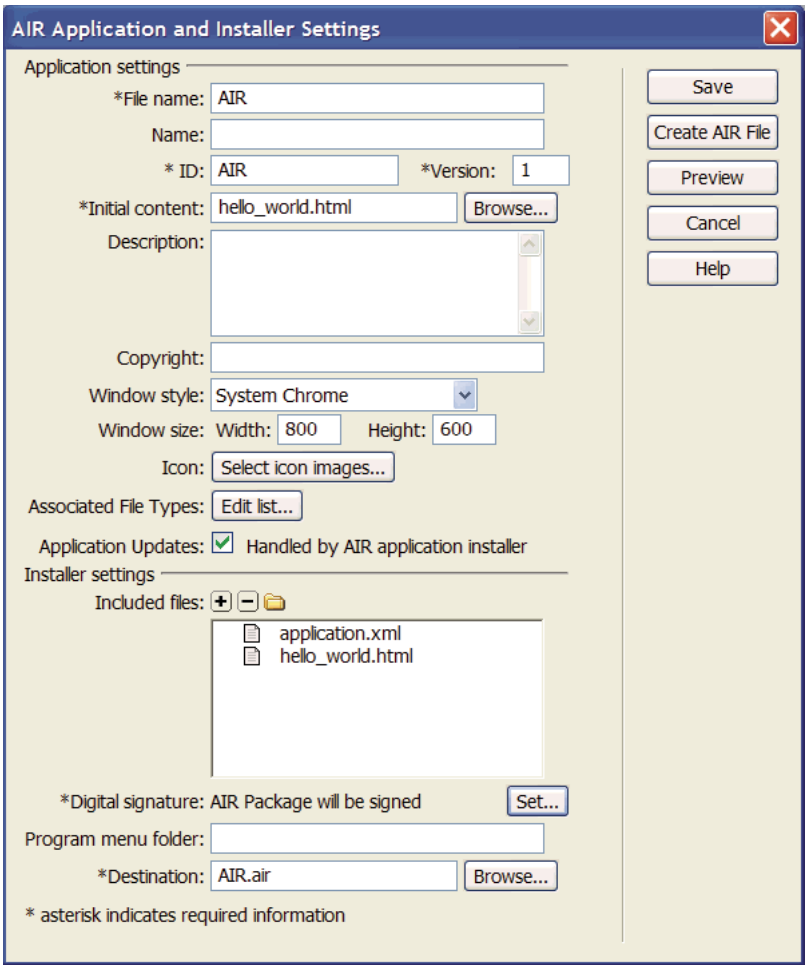
5 En el cuadro de diálogo Firma digital, seleccione Firmar el paquete de AIR con un certificado digital y haga clic en el botón Crear. (Si ya tiene acceso a un certificado digital, puede hacer clic en el botón Examinar para seleccionarlo.)

6 Complete los campos necesarios del cuadro de diálogo Certificado digital con firma automática. Debe indicar su nombre, una contraseña y su confirmación, así como un nombre para el archivo de certificado digital. Dreamweaver guarda el certificado digital en la raíz del sitio.

7 Haga clic en Aceptar para volver al cuadro de diálogo Firma digital.

8 En el cuadro de diálogo Firma digital, indique la contraseña especificada para el certificado digital y haga clic en Aceptar.

El cuadro de diálogo completo AIR - Configuración de aplicación e instalador puede presentar el siguiente aspecto:



Para obtener más información sobre todas las opciones del cuadro de diálogo y cómo modificarlas, consulte Creación de una aplicación de AIR en Dreamweaver.

9 Haga clic en el botón Crear archivo de AIR.

Dreamweaver crea el archivo de la aplicación de Adobe AIR y lo guarda en la carpeta raíz del sitio. Dreamweaver también crea un archivo application.xml y lo guarda en el mismo lugar. Este archivo sirve como manifiesto y define distintas propiedades de la aplicación.

Instalación de la aplicación en el escritorio

Una vez creado el archivo de aplicación, puede instalarlo en cualquier escritorio.

1 Desplace el archivo de aplicación de Adobe AIR fuera del sitio de Dreamweaver y llévelo al escritorio deseado.

   Este paso es opcional. Si lo prefiere, puede instalar la nueva aplicación en el equipo directamente desde el directorio del sitio de Dreamweaver.

2 Haga doble clic en el archivo ejecutable de la aplicación (archivo .air) para instalarla.

Vista previa de una aplicación de Adobe AIR

Se puede obtener una vista previa de las páginas que formarán parte de las aplicaciones de AIR en cualquier momento. Es decir, no necesariamente se debe empaquetar la aplicación antes de ver su aspecto cuando se instale.

1 Compruebe que la página hello_world.html esté abierta en la ventana de documento de Dreamweaver.

2 En la barra de herramientas Documento, haga clic en el botón de previsualización/depuración en el navegador y, a continuación, seleccione Vista previa en AIR.

   También puede presionar Ctrl+Mayús+F12 (Windows) o Cmd+Mayús+F12 (Macintosh).

   Al obtener una vista previa de esta página, fundamentalmente se está viendo lo que vería un usuario como página de inicio de la aplicación una vez instalada en un escritorio.

Capítulo 9: Creación de la primera aplicación de AIR basada en HTML con el SDK de AIR

Para obtener unas indicaciones rápidas y prácticas sobre el funcionamiento de Adobe® AIR®, utilice estas instrucciones para crear y empaquetar una sencilla aplicación "Hello World" de AIR basada en HTML.

Para comenzar, debe tener instalado el motor de ejecución y configurar el SDK de AIR. En este tutorial utilizará *AIR Debug Launcher* (ADL) y *AIR Developer Tool* (ADT). ADL y ADT son programas de utilidades de la línea de comandos que se pueden encontrar en el directorio `bin` del SDK de AIR (consulte "Instalación del SDK de AIR" en la página 21). En este tutorial se asume que está familiarizado con la ejecución de programas desde la línea de comandos y que conoce cómo configurar las variables del entorno de ruta necesarias para el sistema operativo.

Nota: *si es usuario de ® Dreamweaver®, consulte "Creación de la primera aplicación de AIR basada en HTML con Dreamweaver" en la página 39.*

Creación de archivos del proyecto

Todos los proyectos de AIR basados en HTML deben incluir los siguientes archivos: un archivo descriptor de la aplicación, que especifica los metadatos de la aplicación y una página HTML de nivel superior. Además de estos archivos necesarios, este proyecto incluye un archivo de código JavaScript, `AIRAliases.js`, que define las variables de alias adecuadas para las clases de API de AIR.

1. Cree un directorio denominado `HelloWorld` para que incluya los archivos del proyecto.
2. Cree un archivo XML, denominado `HelloWorld-app.xml`.
3. Cree un archivo HTML denominado `HelloWorld.html`.
4. Copie `AIRAliases.js` de la carpeta frameworks del SDK de AIR al directorio project.

Creación del archivo descriptor de la aplicación de AIR

Para comenzar a crear la aplicación de AIR, cree un archivo descriptor de la aplicación XML con la siguiente estructura:

```
<application>
    <id>…</id>
    <version>…</version>
    <filename>…</filename>
    <initialWindow>
        <content>…</content>
        <visible>…</visible>
        <width>…</width>
        <height>…</height>
    </initialWindow>
</application>
```

1 Abra `HelloWorld-app.xml` para la edición.

2 Añada el elemento raíz `<application>`, incluyendo el atributo de espacio de nombres de AIR:

<application xmlns="http://ns.adobe.com/air/application/2.0"> El último segmento del espacio de nombres, "2.0", especifica la versión del motor de ejecución que requiere la aplicación.

3 Añada el elemento `<id>`:

<id>examples.html.HelloWorld</id> El ID de la aplicación la identifica de forma exclusiva junto con el ID de editor (que AIR obtiene del certificado utilizado para firmar el paquete de la aplicación). El ID de la aplicación se utiliza para la instalación, el acceso al directorio privado de almacenamiento del sistema de archivos de la aplicación, el acceso al almacenamiento cifrado privado y la comunicación entre aplicaciones.

4 Añada el elemento `<version>`:

<version>0.1</version> Ayuda a los usuarios a determinar qué versión de la aplicación se está instalando.

5 Agregue el elemento `<filename>`:

<filename>HelloWorld</filename> Nombre utilizado para el ejecutable de la aplicación, el directorio de instalación y otras referencias a la aplicación en el sistema operativo.

6 Añada el elemento `<initialWindow>` que contiene los siguientes elementos secundarios para especificar las propiedades de la ventana de la aplicación inicial:

<content>HelloWorld.html</content> Identifica el archivo HTML raíz para que se cargue AIR.

<visible>true</visible> Hace visible a la ventana de forma inmediata.

<width>400</width> Establece la anchura de la ventana (en píxeles).

<height>200</height> Establece la altura de la ventana.

7 Guarde el archivo. El archivo descriptor de la aplicación completo debe presentar el siguiente aspecto:

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://ns.adobe.com/air/application/2.0">
    <id>examples.html.HelloWorld</id>
    <version>0.1</version>
    <filename>HelloWorld</filename>
    <initialWindow>
        <content>HelloWorld.html</content>
        <visible>true</visible>
        <width>400</width>
        <height>200</height>
    </initialWindow>
</application>
```

En este ejemplo sólo se establecen unas cuantas de las posibles propiedades de la aplicación. Para obtener el conjunto completo de las propiedades de la aplicación, que permiten especificar determinados aspectos, como el tamaño y el fondo cromático de la ventana, la transparencia, el directorio de instalación predeterminado, los tipos de archivo asociados y los iconos de la aplicación, consulte "Configuración de las propiedades de una aplicación de AIR" en la página 71.

Creación de la página HTML de la aplicación

Es necesario crear una sencilla página HTML que sirva como archivo principal para la aplicación de AIR.

1 Abra el archivo `HelloWorld.html` para la edición. Añada el siguiente código HTML:

```
<html>
<head>
    <title>Hello World</title>
</head>
<body onLoad="appLoad()">
    <h1>Hello World</h1>
</body>
</html>
```

2 En la sección `<head>` del HTML, importe el archivo `AIRAliases.js`:

```
<script src="AIRAliases.js" type="text/javascript"></script>
```

AIR define una propiedad denominada `runtime` en el objeto de la ventana HTML. La propiedad runtime proporciona acceso a las clases incorporadas de AIR, utilizando el nombre completo del paquete de la clase. Por ejemplo, para crear un objeto File de AIR se puede añadir la siguiente sentencia en JavaScript:

```
var textFile = new runtime.flash.filesystem.File("app:/textfile.txt");
```

El archivo `AIRAliases.js` define los alias convenientes para las API de AIR más útiles. Con `AIRAliases.js` se puede reducir la referencia a la clase File del siguiente modo:

```
var textFile = new air.File("app:/textfile.txt");
```

3 Bajo la etiqueta de script AIRAliases, añada otra etiqueta de script que contenga una función JavaScript para administrar el evento `onLoad`:

```
<script type="text/javascript">
function appLoad(){
    air.trace("Hello World");
}
</script>
```

La función `appLoad()` simplemente llama a la función `air.trace()`. El mensaje de seguimiento se imprime en la consola de comandos cuando la aplicación se ejecuta utilizando ADL. Las sentencias trace pueden ser muy útiles en la depuración.

4 Guarde el archivo.

El archivo `HelloWorld.html` debe presentar ahora el siguiente aspecto:

```
<html>
<head>
    <title>Hello World</title>
    <script type="text/javascript" src="AIRAliases.js"></script>
    <script type="text/javascript">
        function appLoad(){
            air.trace("Hello World");
        }
    </script>
</head>
<body onLoad="appLoad()">
    <h1>Hello World</h1>
</body>
</html>
```

Prueba de la aplicación

Para ejecutar y probar la aplicación desde la línea de comandos, emplee la utilidad AIR Debug Launcher (ADL). El ejecutable ADL se encuentra en el directorio `bin` del SDK de AIR. Si aún no ha configurado el SDK de AIR, consulte "Instalación del SDK de AIR" en la página 21.

1 Abra una consola de comandos o de shell. Cambie al directorio creado para este proyecto.

2 ejecute el siguiente comando:

```
adl HelloWorld-app.xml
```

Se abrirá una ventana de AIR, mostrando la aplicación. Asimismo, la ventana de la consola muestra el mensaje resultante de la llamada a `air.trace()`.

Para obtener más información, consulte "Configuración de las propiedades de una aplicación de AIR" en la página 71.

Creación de un archivo de instalación de AIR

Cuando la aplicación se ejecute correctamente, puede emplear la utilidad ADT para empaquetar la aplicación en un archivo de instalación de AIR. Un archivo de instalación de AIR contiene todos los archivos de la aplicación, que se pueden distribuir a los usuarios. Se debe instalar Adobe AIR antes de instalar un archivo de AIR empaquetado.

Para garantizar la seguridad de la aplicación, todos los archivos de instalación de AIR se deben firmar digitalmente. Por motivos de desarrollo, se pueden generar certificados básicos con firma automática con ADT u otra herramienta de generación de certificados. También se puede adquirir un certificado con firma de código de una entidad comercial emisora de certificados como, por ejemplo, VeriSign o Thawte. Si los usuarios instalan un archivo de AIR con firma automática, el editor se muestra como "unknown" (desconocido) durante el proceso de instalación. Esto se debe a que el certificado con firma automática sólo garantiza que el archivo de AIR no se ha modificado desde su creación original. No existe ningún método para evitar que alguien firme automáticamente un archivo de AIR de enmascaramiento y lo presente como su aplicación. Para los archivos de AIR distribuidos públicamente, se recomienda el uso de un certificado comercial verificable. Para obtener información general sobre los problemas de seguridad en AIR, consulte Seguridad en AIR (para desarrolladores de ActionScript) o Seguridad en AIR (para desarrolladores de HTML).

Generación de un certificado con firma automática y un par de claves

❖ Desde el símbolo del sistema, indique el siguiente comando (el ejecutable de ADT se ubica en el directorio `bin` del SDK de AIR):

```
adt -certificate -cn SelfSigned 1024-RSA sampleCert.pfx samplePassword
```

ADT genera un archivo de almacén de claves denominado *sampleCert.pfx* que contiene un certificado y la clave privada relacionada.

En este ejemplo se utiliza el número mínimo de atributos que se pueden establecer para un certificado. Se puede utilizar cualquier valor para los parámetros en *cursiva*. El tipo de clave debe ser *1024-RSA* o *2048-RSA* (consulte"Firma digital de archivos de AIR" en la página 96).

Creación de un archivo de instalación de AIR

❖ Desde el símbolo del sistema, introduzca el siguiente comando (en una sola línea):

```
adt -package -storetype pkcs12 -keystore sampleCert.pfx HelloWorld.air
HelloWorld-app.xml HelloWorld.html AIRAliases.js
```

Se le solicitará la contraseña del archivo del almacén de claves.

El argumento HelloWorld.air es el archivo de AIR que genera ADT. HelloWorld-app.xml es el archivo descriptor de la aplicación. Los siguientes argumentos son los archivos utilizados por la aplicación. En este ejemplo sólo se utilizan dos archivos, pero se puede incluir cualquier número de archivos y directorios.

Una vez creado el paquete de AIR, se puede instalar y ejecutar la aplicación haciendo doble clic en el archivo del paquete. También se puede escribir el nombre del archivo de AIR como comando en una ventana de comandos o de shell.

Pasos siguientes

En AIR, el código HTML y JavaScript se suele comportar tal y como lo haría en un navegador web típico. (De hecho, AIR utiliza el mismo motor de representación WebKit que se emplea en el navegador web Safari.) Sin embargo, existen algunas diferencias importantes que se deben conocer a la hora de desarrollar aplicaciones HTML en AIR. Para obtener más información sobre estas diferencias y otros temas importantes, consulte Programming HTML and JavaScript (Programación con HTML y JavaScript; en inglés).

Capítulo 10: Creación de aplicaciones de AIR con las herramientas de la línea de comandos

Las herramientas de la línea de comandos de Adobe® AIR® permiten compilar, probar, empaquetar y firmar aplicaciones de Adobe AIR. Las herramientas de la línea de comandos vienen incluidas en Adobe® Flex™ y el kit de desarrollo de software (SDK) de AIR (http://www.adobe.com/go/learn_air_download_AIRSDK_es).

Los kits de desarrollo de software de AIR y Flex proporcionan las siguientes herramientas de la línea de comandos para desarrollar aplicaciones de AIR:

Compilación Para compilar archivos de origen de Flex MXML y ActionScript, utilice los compiladores *amxmlc* o *acompc*. Para proyectos de Flash Professional, compile el proyecto publicando la película. Las aplicaciones de AIR basadas en HTML no requieren compilación.

Nota: *las herramientas amxmlc y acompc sólo se incluyen en el kit de desarrollo de software (SDK) de Flex y no en el SDK de AIR.*

Pruebas y depuración Utilice *AIR Debug Launcher (ADL)* para probar y depurar aplicaciones de AIR. Se puede ejecutar un proyecto de AIR utilizando ADL sin empaquetar ni instalar la aplicación. ADL imprime el resultado de los errores y los mensajes de seguimiento en la consola de la línea de comandos. ADL también se puede usar para conectarse a la herramienta *Flash Debugger (FDB)* cuando sea necesario depurar más problemas complejos.

Nota: *la herramienta FDB sólo se incluye en el SDK de Flex y no en el SDK de AIR. El SDK de Flex está disponible en http://opensource.adobe.com.*

Empaquetado Utilice *AIR Developer Tool (ADT)* para empaquetar una aplicación completa de AIR en un archivo de instalación de AIR (.air) que se puede instalar en cualquier sistema informático compatible o en un archivo de instalación nativo que sólo se pueda instalar en equipos del mismo tipo que el que ejecuta la herramienta ADT.

Firma Utilice ADT para firmar una aplicación de AIR con un certificado de publicación. Es posible firmar una aplicación al mismo tiempo que se crea el paquete de la aplicación. Asimismo, el empaquetado se puede separar en dos pasos (lo que resulta útil cuando el acceso a los certificados de firma de código y a las claves privadas se encuentra muy controlado). Todas las aplicaciones de AIR se deben firmar.

Nota: *la mayor parte de los entornos de desarrollo integrados, incluyendo Adobe Flash Builder, Adobe Flash Professional y Aptana Studio pueden compilar, depurar y empaquetar aplicaciones de AIR. El uso de las herramientas de la línea de comandos para estas tareas comunes no suele ser necesario cuando ya se utiliza este entorno de desarrollo. No obstante, puede que aún sea necesario emplear estas herramientas de la línea de comandos para las funciones que no se admitan en el entorno de desarrollo integrado. Asimismo, las herramientas de la línea de comandos se pueden utilizar como parte un proceso de creación automatizado.*

Compilación de archivos de origen MXML y ActionScript para AIR

Puede compilar los componentes Adobe® ActionScript® 3.0 y MXML de su aplicación de AIR con el compilador de la línea de comandos MXML (amxmlc). (No es necesario compilar aplicaciones basadas en HTML. Para compilar un archivo SWF en Flash Professional, simplemente publique la película en un archivo SWF.)

El patrón de línea de comandos básico para utilizar amxmlc es:

```
amxmlc [compiler options] -- MyAIRApp.mxml
```

siendo *[compiler options]* las opciones de la línea de comandos que se utilizan para compilar la aplicación de AIR.

El comando amxmlc invoca al compilador de Flex estándar mxmlc con un parámetro adicional, `+configname=air`. Este parámetro indica al compilador que utilice el archivo air-config.xml en lugar de flex-config.xml. Por lo demás, el uso de amxmlc es idéntico al uso de mxmlc. El compilador mxmlc y el formato del archivo de configuración se describen en Creación e implementación de aplicaciones de Flex 3 (en inglés) en la biblioteca de documentación de Flex 3.

El compilador carga el archivo de configuración air-config.xml especificando las bibliotecas de AIR y Flex que se suelen necesitar al compilar una aplicación de AIR. También se puede utilizar un archivo de configuración local a nivel de proyecto para suprimir o añadir opciones adicionales a la configuración global. En general la forma más fácil de crear un archivo de configuración local es mediante modificación de una copia de la versión global. El archivo local puede cargarse con la opción `-load-config`:

-load-config=project-config.xml Suprime las opciones globales.

-load-config+=project-config.xml Añade valores adicionales a las opciones globales que aceptan más de un valor, como la opción `-library-path`. Las opciones globales que tienen un solo valor se suprimen.

Si se utiliza una convención particular para el nombre del archivo de configuración local, el compilador amxmlc carga el archivo local automáticamente. Por ejemplo, si el archivo MXML principal es `RunningMan.mxml`, el nombre del archivo de configuración local es: `RunningMan-config.xml`. Para compilar la aplicación sólo hace falta escribir:

```
amxmlc RunningMan.mxml
```

`RunningMan-config.xml` se carga automáticamente, dado que su nombre de archivo coincide con el del archivo MXML compilado.

Ejemplos con amxmlc

Los siguientes ejemplos demuestran el uso del compilador amxmlc. (Sólo se necesitan compilar los componentes ActionScript y MXML de la aplicación).

Compile un archivo MXML de AIR:

```
amxmlc myApp.mxml
```

Compile y defina el nombre de salida:

```
amxmlc –output anApp.swf -- myApp.mxml
```

Compile un archivo ActionScript de AIR:

```
amxmlc myApp.as
```

Especifique un archivo de configuración para el compilador:

```
amxmlc –load-config config.xml -- myApp.mxml
```

Añada opciones adicionales de otro archivo de configuración:

```
amxmlc –load-config+=moreConfig.xml -- myApp.mxml
```

Añada bibliotecas en la línea de comandos (además de las bibliotecas que ya figuran en el archivo de configuración):

```
amxmlc –library-path+=/libs/libOne.swc,/libs/libTwo.swc  -- myApp.mxml
```

Compile un archivo MXML de AIR sin usar archivo de configuración (Win):

```
mxmlc -library-path [AIR SDK]/frameworks/libs/air/airframework.swc, ^
[AIR SDK]/frameworks/libs/air/airframework.swc, ^
-library-path [Flex 3 SDK]/frameworks/libs/framework.swc ^
-- myApp.mxml
```

Compile un archivo MXML de AIR sin usar archivo de configuración (Mac OS X o Linux):

```
mxmlc -library-path [AIR SDK]/frameworks/libs/air/airframework.swc, \
[AIR SDK]/frameworks/libs/air/airframework.swc, \
-library-path [Flex 3 SDK]/frameworks/libs/framework.swc \
-- myApp.mxml
```

Compile un archivo MXML de AIR para utilizar una biblioteca compartida con el motor de ejecución:

```
amxmlc -external-library-path+=../lib/myLib.swc -runtime-shared-libraries=myrsl.swf --
myApp.mxml
```

Compilando desde Java (con la ruta de clase definida para que incluya `mxmlc.jar`):

```
java flex2.tools.Compiler +flexlib [Flex SDK 3]/frameworks +configname=air [additional
compiler options] -- myApp.mxml
```

La opción flexlib identifica la ubicación del directorio frameworks del SDK de Flex, lo cual permite al compilador localizar el archivo flex_config.xml.

Compilando desde Java (sin ruta de clase definida):

```
java -jar [Flex SDK 2]/lib/mxmlc.jar +flexlib [Flex SDK 3]/frameworks +configname=air
[additional compiler options] -- myApp.mxml
```

Para invocar el compilador utilizando Apache Ant (en el ejemplo se utiliza una tarea de Java para ejecutar mxmlc.jar):

```
<property name="SDK_HOME" value="C:/Flex3SDK"/>
<property name="MAIN_SOURCE_FILE" value="src/myApp.mxml"/>
<property name="DEBUG" value="true"/>
<target name="compile">
    <java jar="${MXMLC.JAR}" fork="true" failonerror="true">
        <arg value="-debug=${DEBUG}"/>
        <arg value="+flexlib=${SDK_HOME}/frameworks"/>
        <arg value="+configname=air"/>
        <arg value="-file-specs=${MAIN_SOURCE_FILE}"/>
    </java>
</target>
```

Compilación de una biblioteca de código o componente de AIR (Flex)

Utilice el compilador de componentes, acompc, para compilar bibliotecas de AIR y componentes independientes. El compilador de componentes acompc se comporta como el compilador amxmlc pero con las siguientes excepciones:

- Es necesario especificar qué clases de las que figuran en el código base se han de incluir en la biblioteca o el componente.
- El compilador acompc no busca automáticamente un archivo de configuración local. Para utilizar un archivo de configuración de un proyecto, hay que utilizar primero la opción –load-config.

El comando acompc invoca el compilador de componentes estándar de Flex, compc, pero carga las opciones de configuración del archivo `air-config.xml` en lugar de las del archivo `flex-config.xml`.

Archivo de configuración del compilador de componentes

Utilice un archivo de configuración local para evitar tener que escribir (quizá con errores tipográficos) la ruta de origin y los nombres de las clases en la línea de comandos. Añada la opción -load-config a la línea de comandos de acompc para cargar el archivo de configuración local.

El siguiente ejemplo ilustra una configuración para crear una biblioteca con dos clases, ParticleManager y Particle, ambos en el paquete `com.adobe.samples.particles`. Los archivos de clase se encuentran en la carpeta `source/com/adobe/samples/particles`.

```
<flex-config>
    <compiler>
        <source-path>
            <path-element>source</path-element>
        </source-path>
    </compiler>
    <include-classes>
        <class>com.adobe.samples.particles.ParticleManager</class>
        <class>com.adobe.samples.particles.Particle</class>
    </include-classes>
</flex-config>
```

Para compilar la biblioteca con el archivo de configuración, denominado `ParticleLib-config.xml`, escriba:

```
acompc -load-config ParticleLib-config.xml -output ParticleLib.swc
```

Para ejecutar el mismo comando con la totalidad en la línea de comandos, escriba:

```
acompc -source-path source -include-classes com.adobe.samples.particles.Particle
com.adobe.samples.particles.ParticleManager -output ParticleLib.swc
```

(Escriba todo el comando en una sola línea o utilice el carácter de continuación de línea para el shell de comandos).

Ejemplos con acompc

Estos ejemplos dan por sentado que utiliza un archivo de configuración denominado `myLib-config.xml`.

Compile un componente o una biblioteca de AIR:

```
acompc -load-config myLib-config.xml -output lib/myLib.swc
```

Compile una biblioteca compartida en tiempo de ejecución

```
acompc -load-config myLib-config.xml -directory -output lib
```

(Obsérvese que la carpeta lib debe existir y estar vacía antes de ejecutar el comando).

Utilización de AIR Debug Launcher (ADL)

Utilice AIR Debug Launcher (ADL) para ejecutar tanto aplicaciones basadas en SWF como las basadas en HTML durante la fase de desarrollo. Con ADL se puede ejecutar una aplicación sin primero tener que empaquetarla e instalarla. De forma predeterminada, ADL utiliza un motor de ejecución incluido con el SDK, con lo cual no se necesita instalar el motor de ejecución por separado para utilizar ADL.

ADL imprime sentencias trace y errores en tiempo de ejecución a la salida estándar, pero no admite puntos de corte u otras funciones de depuración. Flash Debugger (o un entorno de desarrollo integrado como Flash Builder o Aptana Studio) se puede emplear para problemas de depuración complejos.

Inicio de una aplicación con ADL

Para ejecutar una aplicación con ADL, utilice el siguiente patrón:

```
adl application.xml
```

application.xml es el archivo descriptor de la aplicación para ésta.

La sintaxis completa para ADL es la siguiente:

```
adl [-runtime runtime-directory] [-pubid publisher-id] [-nodebug] [-profile profileName]
application.xml [root-directory] [-- arguments]
```

-runtime runtime-directory Especifica el directorio que contiene el motor de ejecución a utilizar. Si no se especifica, se utiliza el directorio del motor de ejecución del mismo SDK que el programa ADL. Si ADL se mueve fuera su carpeta en SDK, especifique el directorio del motor de ejecución. En Windows y Linux, especifique el directorio que contiene el directorio `Adobe AIR`. En Mac OS X, especifique el directorio que contiene `Adobe AIR.framework`.

-pubid publisher-id Asigna el valor especificado como ID del editor de la aplicación de AIR para esta ejecución. La especificación de un ID de editor temporal permite ensayar las funciones de una aplicación de AIR, como la comunicación a través de una conexión local, que utilizan el ID del editor para ayudar a identificar una aplicación con exclusividad. A partir de AIR 1.5.3, también se puede especificar el ID de editor en el archivo descriptor de la aplicación (y no se debe utilizar este parámetro).

Nota: *a partir de AIR 1.5.3, un ID de editor no se vuelve a calcular ni a asignar automáticamente a una aplicación de AIR. Se puede especificar un ID de editor al crear una actualización en una aplicación de AIR existente, pero las nuevas aplicaciones no necesitan ni deben especificar un ID de editor.*

-nodebug Desactiva la compatibilidad con la depuración. Si se utiliza, el proceso de la aplicación no podrá conectar con el depurador de Flash y se suprimen los cuadros de diálogo para excepciones no controladas. (Sin embargo, las sentencias trace continúan imprimiéndose en la ventana de la consola.) Si desactivamos la depuración, el funcionamiento de la aplicación se agilizará y el modo de ejecución será más similar al de una aplicación instalada.

-profile profileName ADL depura la aplicación utilizando el perfil especificado. *`profileName`* puede ser `desktop`, `extendedDesktop`, `mobileDevice` y `extendedMobileDevice`. Para obtener más información, consulte "Limitación de los perfiles de la aplicación de destino" en la página 77 y "Perfiles de la aplicación" en la página 83.

application.xml Archivo descriptor de la aplicación. Consulte "Configuración de las propiedades de una aplicación de AIR" en la página 71. El descriptor de la aplicación es el único parámetro que requiere ADL y, en la mayoría de los casos, el único parámetro necesario.

root-directory Especifica el directorio raíz de la aplicación a ejecutar. Si no se especifica, se utilizará el directorio que contiene el archivo descriptor de la aplicación.

-- **arguments** Las cadenas de caracteres que aparezcan después de "--" se pasan a la aplicación como argumentos de la línea de comandos.

Nota: *cuando se intenta iniciar una aplicación de AIR que ya está ejecutándose, no se inicia una nueva instancia de la aplicación, sino que se distribuye un evento* `invoke` *a la instancia que está en ejecución.*

Impresión de sentencias trace

Para imprimir sentencias trace en la consola que se utiliza para ejecutar ADL, añada sentencias trace al código con la función `trace()`:

Ejemplo de JavaScript:

```
//ActionScript
trace("debug message");
```

Ejemplo de ActionScript:

```
//JavaScript
air.trace("debug message");
```

En el código JavaScript se pueden utilizar las funciones `alert()` y `confirm()` para mostrar los mensajes de depuración de la aplicación. Además, los números de línea para los errores de sintaxis, así como las excepciones de JavaScript sin capturar se imprimen en la consola.

Nota: *para utilizar el prefijo air que muestra en el ejemplo de JavaScript, debe importar el archivo AIRAliases.js en la página. El archivo se ubica en el directorio frameworks del SDK de AIR.*

Ejemplos con ADL

Ejecute una aplicación del directorio actual:

```
adl myApp-app.xml
```

Ejecute una aplicación de un subdirectorio del directorio actual:

```
adl source/myApp-app.xml release
```

Ejecute una aplicación y transmita dos argumentos de la línea de comandos, "tick" y "tock":

```
adl myApp-app.xml -- tick tock
```

Ejecute una aplicación con un motor de ejecución específico:

```
adl -runtime /AIRSDK/runtime myApp-app.xml
```

Ejecute una aplicación sin compatibilidad de depuración:

```
adl myApp-app.xml -nodebug
```

Ejecute una aplicación utilizando Apache Ant para ejecutarla:

```
<property name="SDK_HOME" value="C:/AIRSDK"/>
<property name="ADL" value="${SDK_HOME}/bin/adl.exe"/>
<property name="APP_DESCRIPTOR" value="$src/myApp-app.xml"/>

<target name="test">
    <exec executable="${ADL}">
        <arg value="${APP_DESCRIPTOR}"/>
    </exec>
</target>
```

Conexión a Flash Debugger (FDB)

Para depurar aplicaciones de AIR con Flash Debugger, inicie una sesión de FDB y posteriormente inicie la aplicación con ADL.

Nota: *en las aplicaciones de AIR basadas en SWF los archivos de origen de ActionScript se deben compilar con el indicador* `-debug`*. (En Flash Professional, seleccione la opción Permitir depuración en el cuadro de diálogo Configuración de publicación.)*

1. Inicie FDB. El programa FDB se encuentra en el directorio `bin` del SDK de Flex.

   La consola presenta el indicador de FDB: `<fdb>`

2. Ejecute el comando `run`: `<fdb>run [Enter]`
3. En otra consola de comandos o de shell, inicie una versión de depuración de la aplicación:

   ```
   adl myApp.xml
   ```

4. Utilice comandos de FDB para definir los puntos de corte según proceda.
5. Escriba:: `continue [Intro]`

Si una aplicación de AIR se basa en SWF, el depurador sólo controla la ejecución de código ActionScript. Si la aplicación de AIR se basa en HTML, el depurador sólo controla la ejecución de código JavaScript.

Para ejecutar ADL sin conexión con el depurador, incluya la opción `-nodebug`:

```
adl myApp.xml -nodebug
```

Para obtener información básica sobre los comandos de FDB, ejecute el comando`help`:

```
<fdb>help [Enter]
```

Para obtener información sobre los comandos de FDB, consulte Using the command-line debugger commands (Uso de los comandos del depurador de la línea de comandos) en la documentación de Flex (en inglés).

Códigos de error y de salida de ADL

En el cuadro siguiente se describen los códigos de error o de salida que imprime ADL:

Código de salida	Descripción
0	Inicio satisfactorio. ADL se cierra después de cerrarse la aplicación de AIR.
1	Invocación satisfactoria de una aplicación de AIR que ya está en ejecución. ADL se cierra inmediatamente.
2	Error de uso. Los argumentos proporcionados a ADL son incorrectos.
3	No se puede encontrar el motor de ejecución.

Código de salida	Descripción
4	No se puede iniciar el motor de ejecución. Esto se debe con frecuencia a que la versión o el nivel de revisión que se especifica en la aplicación no coincide con la versión o el nivel de revisión del motor de ejecución.
5	Se ha producido un error de causa desconocida.
6	No se puede encontrar el archivo descriptor de la aplicación.
7	El contenido del descriptor de la aplicación no es válido. Este error suele indicar que el XML no está bien conformado.
8	No se puede encontrar el archivo de contenido principal de la aplicación (especificado en el elemento <content> del archivo descriptor de la aplicación).
9	El archivo de contenido principal de la aplicación no es un archivo SWF o HTML válido.

Empaquetado de archivos de instalación de AIR con AIR Developer Tool (ADT)

El archivo de instalación de AIR, tanto para las aplicaciones de AIR basadas en SWF como para las basadas en HTML, se crean con AIR Developer Tool (ADT).

ADT es un programa en Java que se puede ejecutar desde la línea de comandos o una herramienta de construcción como Ant. El kit de desarrollo de software incluye scripts de la línea de comandos que ejecutan el programa Java.

Si utiliza Flash Builder para crear la aplicación, también puede utilizar el asistente de exportación de Flash Builder para crear el paquete de archivos de AIR. Consulte Developing AIR applications with Flash Builder (Desarrollo de aplicaciones de AIR con Flash Builder).

Si está utilizando Flash Professional CS5, utilice el comando Archivo > Publicar para crear el paquete de AIR. Para obtener más información, consulte Publicación para Adobe AIR en Uso de Flash (CS5). Si se está utilizando Adobe Flash CS3 o CS4 Professional, puede utilizar Comandos > Crear archivo de AIR para crear el paquete de AIR. Para obtener más información, consulte Publicación para Adobe AIR en Uso de Flash (CS4).

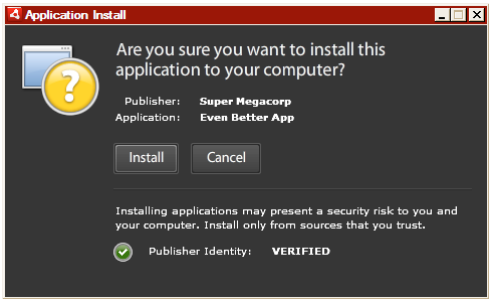
Si utiliza Adobe® AIR™ Extension para Dreamweaver® para crear la aplicación, también puede emplear el comando Crear archivo de AIR para crear el paquete de AIR. Este comando se encuentra en el cuadro de diálogo AIR - Configuración de aplicación e instalador.

Empaquetado de archivos de instalación de AIR

Cada aplicación de AIR debe incluir, como mínimo, un archivo descriptor de la aplicación y un archivo SWF o HTML principal. Cualquier otro recurso que se vaya a instalar con la aplicación se debe empaquetar también en el archivo de AIR.

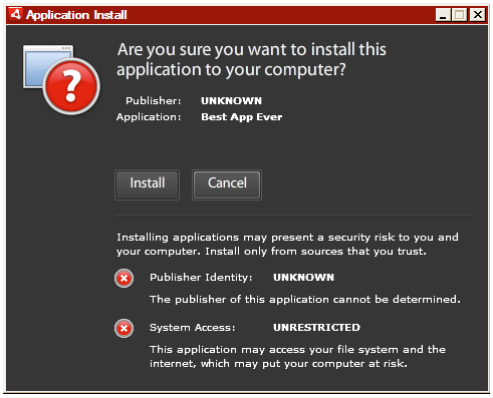
Todos los archivos instaladores de AIR deben firmarse con un certificado digital. El instalador de AIR utiliza la firma para verificar que el archivo de la aplicación no ha sido modificado desde que se firmó. Puede utilizar un certificado de firma de código de una entidad emisora de certificados o un certificado con firma automática.

Cuando se utiliza un certificado emitido por una autoridad de certificación de confianza, se ofrece a los usuarios de la aplicación cierta seguridad respecto de su identidad como editor de la aplicación. El cuadro de diálogo de instalación refleja el hecho de que su identidad se verifica mediante la entidad emisora de certificados:



Cuadro de diálogo de confirmación de instalación firmado por un certificado de confianza

Cuando se utiliza un certificado con firma automática, los usuarios no pueden verificar su identidad como firmante. Un certificado con firma automática también debilita la garantía de que el paquete no se haya modificado. (Esto se debe a que un archivo de instalación de confianza se puede sustituir por una falsificación antes de que llegue al usuario.) El cuadro de diálogo de instalación refleja que la identidad del editor no puede comprobarse. Los usuarios están arriesgando más su seguridad cuando instalan la aplicación:



Cuadro de diálogo de confirmación de instalación firmado por un certificado con firma automática

Se puede empaquetar y firmar un archivo de AIR en un solo paso con el comando `-package` de ADT. También se puede crear un paquete intermedio sin firmar con el comando `-prepare`, firmando después el paquete intermedio con el comando `-sign` en un paso separado.

Nota: *las versiones de Java 1.5 y posteriores no aceptan caracteres ASCII superior en contraseñas utilizadas para proteger archivos de certificado PKCS12. Cuando se cree o se exporte el archivo de certificado de firma de código, utilice solamente caracteres ASCII normales en la contraseña.*

Al firmar el paquete de instalación, ADT se pone en contacto automáticamente con el servidor de una autoridad de marcas de hora para verificar la hora. La marca de hora se incluye en el archivo de AIR. Un archivo de AIR que incluya una marca de hora verificada podrá instalarse en el futuro en cualquier momento. Si ADT no puede conectarse al servidor de marcas de hora, se cancela el empaquetado. La opción de marca de hora puede pasarse por alto, pero sin marca de hora la aplicación de AIR ya no podrá instalarse una vez caducado el certificado que se utilizó para firmar el archivo de instalación.

Si se está creando un paquete para actualizar una aplicación de AIR existente, el paquete se debe firmar con el mismo certificado que la aplicación original. Si el certificado original se ha renovado o ha caducado en los últimos 180 días, o bien, si se desea cambiar a un nuevo certificado, se puede aplicar una firma de migración. La firma de migración implica firmar el archivo de AIR de la aplicación tanto con el nuevo como con el antiguo certificado. Utilice el comando `-migrate` para aplicar la firma de migración tal y como se describe en "Firma de un archivo de AIR para cambiar el certificado de la aplicación" en la página 68.

Importante: *existe un estricto período de gracia de 180 días para aplicar una firma de migración una vez caducado el certificado original. Sin la firma de migración, los usuarios existentes deben desinstalar su aplicación existente antes de instalar la nueva versión. El período de gracia de días sólo se aplica a las aplicaciones que especifican la versión de AIR 1.5.3, o superior, en el espacio de nombres del descriptor de la aplicación. No hay período de gracia cuando se utilizan versiones anteriores del motor de ejecución de AIR.*

Antes de AIR 1.1, no se admitían las firmas de migración. La aplicación se debe empaquetar con un SDK de versión 1.1 o posterior para aplicar una firma de migración.

Las aplicaciones implementadas utilizando archivos de AIR se denominan aplicaciones de perfil de escritorio. No es posible utilizar ADT para empaquetar un instalador nativo para una aplicación de AIR si el archivo descriptor de la aplicación no admite el perfil de escritorio. Este perfil se puede restringir utilizando el elemento supportedProfiles en el archivo descriptor de la aplicación. Consulte "Limitación de los perfiles de la aplicación de destino" en la página 77.

Nota: *los valores definidos en el archivo descriptor de la aplicación determinan la identidad de la aplicación de AIR y su ruta de instalación predeterminada. Consulte "Estructura del archivo descriptor de la aplicación" en la página 71.*

IDs de editor

A partir de AIR 1.5.3, los IDs de editor quedan desfasados. Las nuevas aplicaciones (publicadas en un principio con AIR 1.5.3 o superior) no necesitan ni deben especificar un ID de editor.

Al actualizar las aplicaciones publicadas con versiones anteriores de AIR, se debe especificar el ID de editor original en el archivo descriptor de la aplicación. De lo contrario, la versión instalada de la aplicación y la versión de actualización se tratan como aplicaciones diferentes. Si se utiliza un ID distinto o se omite la etiqueta publisherID, un usuario debe desinstalar la versión anterior antes de instalar la nueva.

Para determinar el ID de editor original, localice el archivo `publisherid` en el subdirectorio META-INF/AIR donde se instaló la aplicación original. La cadena de este archivo es el ID de editor. El descriptor de la aplicación debe especificar el motor de ejecución de AIR 1.5.3 (o posterior) en la declaración del espacio de nombres del archivo descriptor de la aplicación con el fin de especificar el ID de editor manualmente.

Para las aplicaciones publicadas antes de AIR 1.5.3, o que se publican con el SDK de AIR 1.5.3, al especificar una versión anterior de AIR en el espacio de nombres del descriptor de la aplicación, se calcula un ID de editor en función del certificado de firma. Este ID se utiliza, junto con el ID de la aplicación, para determinar la identidad de una aplicación. El ID de editor, cuando se encuentra presente, se emplea para lo siguiente:

- Comprobar que un archivo de AIR es una actualización en lugar de una nueva aplicación para instalar.
- Como parte de la clave de cifrado para el almacén local cifrado.
- Como parte de la ruta para el directorio de almacenamiento de la aplicación.
- Como parte de la cadena de conexión para conexiones locales.
- Como parte de la cadena de identidad utilizada para invocar una aplicación la API en navegador de AIR.
- Como parte de OSID (utilizado al crear programas personalizados de instalación y desinstalación).

Antes de AIR 1.5.3, el ID de editor de una aplicación podía cambiar si se firmaba una actualización de la aplicación con una firma de migración utilizando un certificado nuevo o renovado. Cuando cambia un ID de editor, el comportamiento de cualquier función de AIR basada en el ID también cambia. Por ejemplo, ya no se puede acceder a los datos del almacén local cifrado existente y cualquier instancia de Flash o AIR que cree una conexión local con la aplicación debe utilizar el nuevo ID en la cadena de conexión.

En AIR 1.5.3 o posterior, el ID de editor no se basa en el certificado de firma y sólo se asigna si la etiqueta publisherID se incluye en el descriptor de la aplicación. Una aplicación no puede actualizarse si el ID de editor especificado para el paquete de AIR de actualización no coincide con su ID de editor actual.

Empaquetado y firma de un archivo de AIR en un solo paso

❖ Utilice el comando `-package` con la siguiente sintaxis (en una sola línea de comandos):

```
adt -package SIGNING_OPTIONS air_file app_xml [file_or_dir | -C dir file_or_dir | -e file dir ...] ...
```

SIGNING_OPTIONS Las opciones de firma identifican al almacén de claves que contiene la clave privada y el certificado que se utilizaron para firmar el archivo de AIR. Para firmar una aplicación de AIR con un certificado con firma automática generado por ADT, las opciones a utilizar son:

```
-storetype pkcs12 -keystore certificate.p12
```

En este ejemplo, *certificate.p12* es el nombre del archivo del almacén de claves. (ADT le solicita la contraseña, dado que no se indica en la línea de comandos). Las opciones de firma se describen con detenimiento en "Opciones de firma en la línea de comandos de ADT" en la página 63.

air_file El nombre del archivo de AIR que se va a crear.

app_xml La ruta al archivo descriptor de la aplicación. La ruta se puede indicar en relación con el directorio actual o como ruta absoluta. (En el archivo de AIR el archivo descriptor de la aplicación cambia de nombre a "application.xml").

file_or_dir Los archivos y directorios a empaquetar en el archivo de AIR. Se puede especificar la cantidad de archivos y directorios que se desee, delimitados por un espacio en blanco. Si se incluye un directorio en la lista, se añaden al paquete todos los archivos y subdirectorios que contenga, excepto los archivos ocultos. (Si se especifica el archivo descriptor de la aplicación, sea directamente o mediante el uso de un comodín o por expansión de un directorio, éste se pasa por alto y no se añade al paquete por segunda vez). Los archivos y directorios especificados deben estar incluidos en el directorio actual o en uno de sus subdirectorios. Para cambiar el directorio actual, utilice la opción `-C`.

Importante: *no se pueden utilizar comodines en los argumentos* `file_or_dir` *después de la opción* `-C`*. (Los shells de comandos expanden los comodines antes de pasar los argumentos a ADT, por lo que ADT buscaría los archivos en el lugar equivocado). Sí se puede utilizar el punto (".") para representar el directorio actual. Por ejemplo:* `"-C assets ."` *copia todo el contenido del directorio assets, incluidos los subdirectorios, hasta el nivel raíz del paquete de la aplicación.*

`-C dir` Cambia el directorio de trabajo al valor *dir* antes de procesar los archivos y directorios posteriores que se añadan al paquete de la aplicación. Los archivos o directorios se añaden a la raíz del paquete de la aplicación. La opción `-C` se puede utilizar todas las veces que se desee para incluir archivos desde varios lugares del sistema de archivos. Si se especifica una ruta relativa para `dir`, la ruta siempre se resuelve desde el directorio de trabajo original.

A medida que ADT procesa los archivos y directorios incluidos en el paquete, se van guardando las rutas relativas entre el directorio actual y los archivos de destino. Estas rutas se expanded en la estructura de directorios de la aplicación cuando se instala el paquete. Por lo tanto, si se especifica `-C release/bin lib/feature.swf`, el archivo `release/bin/lib/feature.swf` se coloca en el subdirectorio `lib` de la carpeta raíz de la aplicación.

`-e file dir` Coloca el archivo especificado en el directorio especificado del paquete.

Nota: *el elemento* `<content>` *del archivo descriptor de la aplicación debe especificar la ubicación final del archivo principal de la aplicación dentro del árbol de directorios del paquete de la aplicación.*

También se puede empaquetar y distribuir una aplicación de AIR utilizando un instalador nativo (por ejemplo, un archivo EXE en Windows o un archivo DMG en Mac OS). Para obtener más información, consulte "Empaquetado de una aplicación de AIR en un instalador nativo" en la página 65.

Ejemplos del comando ADT -package

Empaquete los archivos de la aplicación específicos en el directorio actual para una aplicación de AIR basada en SWF:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf components.swc
```

Empaquete los archivos de la aplicación específicos en el directorio actual para una aplicación de AIR basada en HTML:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.html AIRAliases.js
image.gif
```

Empaquete todos los archivos y subdirectorios del directorio de trabajo actual:

```
adt -package -storetype pkcs12 -keystore ../cert.p12 myApp.air myApp.xml .
```

Nota: *el archivo del almacén de claves contiene la clave privada que se utilizó para firmar la aplicación. No incluya nunca el certificado de firma en el paquete de AIR. Si utiliza comodines en el comando ADT, coloque el archivo del almacén de claves en otra carpeta para que no se incluya en el paquete. En este ejemplo el archivo del almacén de claves, cert.p12, reside en el directorio principal.*

Empaquete solamente los principales archivos y un subdirectorio de imágenes:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml myApp.swf images
```

Empaquete una aplicación basada en HTML y todos los archivos de los subdirectorios HTML, scripts e images:

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml index.html AIRALiases.js
html scripts images
```

Empaquete el archivo application.xml y el archivo SWF principal que se encuentran en un directorio de trabajo (release/bin):

```
adt -package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml -C
release/bin myApp.swf
```

Empaquete componentes procedentes de más de un lugar en el sistema de archivos de construcción. En este ejemplo, los componentes de la aplicación se encuentran en las siguientes carpetas antes de que se empaquetan:

```
/devRoot
    /myApp
        /release
            /bin
                myApp.xml
                myApp.swf or myApp.html
    /artwork
        /myApp
            /images
                image-1.png
                ...
                image-n.png
    /libraries
        /release
            /libs
                lib-1.swf
                lib-2.swf
                lib-a.js
                AIRAliases.js
```

La ejecución del siguiente comando ADT desde el directorio /devRoot/myApp:

```
adt –package -storetype pkcs12 -keystore cert.p12 myApp.air release/bin/myApp.xml
    –C release/bin myApp.swf (or myApp.html)
    –C ../artwork/myApp images
    –C ../libraries/release libs
```

produce un paquete con la siguiente estructura:

```
/myAppRoot
    /META-INF
        /AIR
            application.xml
            hash
    myApp.swf
    mimetype
    /images
        image-1.png
        ...
        image-n.png
    /libs
        lib-1.swf
        lib-2.swf
        lib-a.js
        AIRAliases.js
```

Ejecute ADT como programa Java para una aplicación sencilla basada en SWF (sin establecer la ruta de clase):

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.swf
```

Ejecute ADT como programa Java para una aplicación sencilla basada en HTML (sin establecer la ruta de clase):

```
java –jar {AIRSDK}/lib/ADT.jar –package -storetype pkcs12 -keystore cert.p12 myApp.air
myApp.xml myApp.html AIRAliases.js
```

Ejecute ADT como programa de Java (con la ruta de clase de Java definida para incluir el paquete ADT.jar):

```
java com.adobe.air.ADT –package -storetype pkcs12 -keystore cert.p12 myApp.air myApp.xml
myApp.swf
```

Ejecute ADT como tarea de Java en Apache Ant:

```
<property name="SDK_HOME" value="C:/AIRSDK"/>
<property name="ADT.JAR" value="${SDK_HOME}/lib/adt.jar"/>

target name="package">
    <java jar="${ADT.JAR}" fork="true" failonerror="true">
        <arg value="-package"/>
        <arg value="-storetype"/>
        <arg value="pkcs12"/>
        <arg value="-keystore"/>
        <arg value="../../ExampleCert.p12"/>
        <arg value="myApp.air"/>
        <arg value="myApp.xml"/>
        <arg value="myApp.xml"/>
        <arg value="icons/*.png"/>
    </java>
</target>
```

Mensajes de error de ADT

La siguiente tabla incluye los posibles errores que puede notificar el programa ADT y las causas probables:

Errores de validación del descriptor de la aplicación

Código de error	Descripción	Notas
100	El descriptor de la aplicación no se puede analizar.	Compruebe el archivo descriptor de la aplicación para buscar errores de sintaxis XML como, por ejemplo, etiquetas sin cerrar.
101	Falta el espacio de nombres.	Añada el espacio de nombres ausente.
102	Espacio de nombres no válido.	Compruebe la ortografía del espacio de nombres.
103	Elemento o atributo inesperado	Elimine los elementos y atributos erróneos. Los valores personalizados no se permiten en el archivo descriptor. Compruebe la ortografía de los nombres de atributos y elementos. Asegúrese de que los elementos se sitúen en el elemento principal correcto y que los atributos se utilicen con los elementos adecuados.
104	Falta un elemento o atributo	Añada el elemento o atributo necesario.

Código de error	Descripción	Notas
105	El elemento o atributo contienen un valor no válido.	Corrija el valor erróneo.
106	Combinación de atributo de ventana no válida.	Algunas configuraciones de ventana como, por ejemplo, `transparency = true` y `systemChrome = standard` no se pueden utilizar de forma conjunta. Cambie una de las configuraciones incompatibles.
107	El tamaño mínimo de ventana es superior al tamaño máximo de ventana.	Modifique la configuración de tamaño máximo o mínimo.

Consulte "Configuración de las propiedades de una aplicación de AIR" en la página 71 para obtener información sobre los espacios de nombres, elementos, atributos y sus valores válidos.

Errores de icono de la aplicación

Código de error	Descripción	Notas
200	El archivo de icono se puede abrir.	Compruebe que el archivo existe en la ruta especificada. Utilice otra aplicación para garantizar que el archivo se puede abrir.
201	El tamaño del icono no es correcto.	El tamaño del icono (en píxeles) debe coincidir con la etiqueta XML. Por ejemplo, con el siguiente elemento descriptor de la aplicación: `<image32x32>icon.png</image32x32>` La imagen en `icon.png` debe ser exactamente de 32x32 píxeles.
202	El archivo de icono contiene un formato de imagen no compatible.	Sólo se admite el formato PNG. Convierta imágenes en otros formatos antes de empaquetar la aplicación.

Errores de archivo de aplicación

Código de error	Descripción	Notas
300	Falta el archivo o no se puede abrir.	No se puede encontrar o no se puede abrir un archivo especificado en la línea de comandos.
301	Falta el archivo descriptor de la aplicación o no se puede abrir.	El archivo descriptor de la aplicación no se encuentra en la ruta especificada o no puede abrirse.
302	Falta el archivo de contenido raíz en un paquete.	El archivo SWF o HTML al que se hace referencia en el elemento `<content>` del descriptor de la aplicación se debe añadir al paquete agregándolo a los archivos que se incluyen en la línea de comandos de ADT.

Código de error	Descripción	Notas
303	Falta el archivo de icono en el paquete.	Los archivos de icono especificado en el descriptor de la aplicación se deben añadir al paquete agregándolos a los archivos que se incluyen en la línea de comandos de ADT. Los archivos de icono no se agregan automáticamente.
304	El contenido de ventana inicial no es válido.	El archivo al que se hace referencia en el elemento `<content>` del descriptor de la aplicación no se reconoce como archivo HTML o SWF válido.
305	La versión de SWF del contenido de ventana inicial sobrepasa la versión del espacio de nombres.	La versión de SWF del archivo al que se hace referencia en el elemento `<content>` del descriptor de la aplicación no se admite en la versión de AIR especificada en el espacio de nombres del descriptor. Por ejemplo, si se intenta empaquetar un archivo SWF10 (Flash Player 10) como contenido inicial de una aplicación de AIR 1.1, se generará este error.
306	Perfil no admitido.	El perfil que está especificando en el archivo descriptor de la aplicación no se admite. Consulte "Limitación de los perfiles de la aplicación de destino" en la página 77.
	El espacio de nombres no admite funciones.	Utilice el espacio de nombres apropiado para las funciones (por ejemplo, el espacio de nombres 2.0).

Códigos de salida para otros errores

Código de salida	Descripción	Notas
2	Error de uso	Busque errores en los argumentos de la línea de comandos.
5	Error desconocido.	Este error indica una situación que no se puede explicar con condiciones de error comunes. Entre las posibles causas de raíz se incluyen la incompatibilidad entre ADT y el entorno de ejecución de Java (JRE), instalaciones dañadas de ADT o JRE y errores de programación en ADT.
6	No se puede en el directorio de salida.	Compruebe que se pueda acceder al directorio de salida especificado (o implicado) y que la unidad que lo contiene tenga suficiente espacio en disco.
7	No se puede acceder al certificado.	Compruebe que la ruta al almacén de claves se especifique correctamente. Compruebe que se pueda acceder al certificado del almacén de claves. La utilidad Java 1.6 Keytool se puede utilizar para ayudar a solucionar problemas de acceso a certificados.

Código de salida	Descripción	Notas
8	Certificado no válido.	El archivo del certificado tiene un formato incorrecto, se ha modificado, ha caducado o se ha revocado.
9	No se pudo firmar un archivo de AIR	Compruebe las opciones de firma transmitidas a ADT.
10	No se ha podido crear la marca de hora.	ADT no ha podido establecer una conexión con el servidor de marcas de hora. Si se conecta a Internet mediante un servidor proxy, puede que sea necesario establecer la configuración de proxy de JRE.
11	Error de creación del certificado.	Compruebe los argumentos de la línea de comandos para crear firmas.
12	Entrada no válida.	Compruebe las rutas de archivo y otros argumentos transmitidos a ADT en la línea de comandos.

Opciones de firma en la línea de comandos de ADT

ADT utiliza Java Cryptography Architecture (JCA) para acceder a las claves privadas y los certificados para firmar aplicaciones de AIR. Las opciones de firma identifican el almacén de claves, así como la clave privada y el certificado que en él se encuentran.

El almacén de claves debe incluir tanto la clave privada como la cadena de certificación asociada. Si el certificado de firma está encadenado con un certificado de confianza en un ordenador, el nombre común del certificado se presenta como el nombre del editor en el cuadro de diálogo de la instalación de AIR.

ADT exige que el certificado cumpla con la norma x509v3, RFC3280 (en inglés), e incluya la extensión de uso mejorado de claves con los valores correspondientes para la firma de códigos. Se respetan las limitaciones propias del certificado y éstas podrían descartar el uso de algunos certificados para firmar aplicaciones de AIR.

Nota: *ADT utiliza la configuración de proxy para el entorno de ejecución de Java (JRE), si procede, para conectarse a recursos de Internet con la finalidad de comprobar listas de revocación de certificados y obtener marcas de hora. Si tiene algún problema con la conexión a recursos de Internet cuando utiliza ADT y la red requiere una configuración de proxy concreta, es posible que necesite ajustar la configuración de proxy para JRE.*

Especificación de las opciones de firma de AIR

❖ Para especificar las opciones de firma de ADT para los comandos `-package` y `-prepare`, utilice la sintaxis siguiente:

```
[-alias aliasName] [-storetype type] [-keystore path] [-storepass password1] [-keypass password2] [-providerName className] [-tsa url]
```

-alias aliasName : alias de una clave en el almacén de claves. No es necesario especificar un alias si el almacén de claves contiene un solo certificado. Si no se especifica ningún alias, ADT utiliza la primera clave del almacén de claves.

No todas las aplicaciones con administración del almacén de claves permiten que se asigne un alias a los certificados. Si hace uso del almacén de claves del sistema en Windows, por ejemplo, utilice como alias el nombre distinguido del certificado. Se puede utilizar la utilidad Java Keytool para enumerar los certificados disponibles para poder determinar el alias. Por ejemplo, si se ejecuta el comando:

```
keytool -list -storetype Windows-MY
```

se produce una salida para un certificado como la siguiente:

```
CN=TestingCert,OU=QE,O=Adobe,C=US, PrivateKeyEntry,
Certificate fingerprint (MD5): 73:D5:21:E9:8A:28:0A:AB:FD:1D:11:EA:BB:A7:55:88
```

Para hacer referencia a este certificado en la línea de comandos de ADT, defina el alias en:

```
CN=TestingCert,OU=QE,O=Adobe,C=US
```

En Mac OS X, el alias de un certificado en la Llave es el nombre que se muestra en la aplicación Acceso a Llaveros.

-storetype type : tipo de almacén de claves, determinado por la implementación del almacén de claves. La implementación del almacén de claves predeterminada incluida con la mayoría de las instalaciones de Java es compatible con los tipos `JKS` y `PKCS12`. Java 5.0 ofrece compatibilidad con el tipo `PKCS11`, para acceder a almacenes de claves en tokens de hardware, y al tipo `Keychain`, para acceder a la Llave de Mac OS X. Java 6.0 ofrece compatibilidad con el tipo `MSCAPI` (en Windows). Si se han instalado y configurado otros proveedores de JCA, puede que se disponga además de otros tipos de almacenes de claves. Si no se especifica ningún tipo de almacén de claves, se utiliza el tipo predeterminado para el proveedor de JCA predeterminado.

Tipo de almacén	Formato del almacén de claves	Versión mínima de Java
JKS	Archivo de almacén de claves de Java (.keystore)	1.2
PKCS12	Archivo PKCS12 (.p12 o .pfx)	1.4
PKCS11	Token de hardware	1.5
KeychainStore	Llave de Mac OS X	1.5
Windows-MY o Windows-ROOT	MSCAPI	1.6

-keystore path: ruta al archivo del almacén de claves para tipos de almacén basados en archivo.

-storepass password1: contraseña para acceder al almacén de claves. Si no se especifica, ADT la solicita.

-keypass password2: contraseña para acceder a la clave privada que se utiliza para firmar la aplicación de AIR. Si no se especifica, ADT la solicita.

-providerName className: proveedor de JCA para el tipo de almacén de claves especificado. Si no se especifica, ADT utiliza el proveedor predeterminado para ese tipo de almacén de claves.

-tsa url: especifica la URL de un RFC3161 (en inglés) para marcar la hora en la firma digital. Si no se especifica una URL, se utiliza un servidor de marcas de hora predeterminado suministrado por Geotrust. Cuando la firma de una aplicación de AIR lleva una marca de hora, la aplicación puede instalarse después de caducado el certificado de firma porque la marca de hora verifica que el certificado era válido al momento de firmar.

Si ADT no puede conectarse al servidor de marcas de hora, se cancela la firma y no se produce ningún paquete. La marca de hora se puede desactivar especificando `-tsa none`. Sin embargo, una aplicación de AIR empaquetada sin marca de hora no se puede instalar una vez caducado el certificado de firma.

Nota: *las opciones de firma son como las opciones equivalentes de la utilidad Keytool de Java. La utilidad Keytool sirve para examinar y administrar almacenes de claves en Windows. La utilidad de seguridad de Apple® sirve para lo mismo en Mac OS X.*

Ejemplos de opciones de firma

Firma con un archivo .p12:

```
-storetype pkcs12 -keystore cert.p12
```

Firma con el almacén de claves de Java predeterminado:

```
-alias AIRcert -storetype jks
```

Firma con un almacén de claves de Java específico:

```
-alias AIRcert -storetype jks -keystore certStore.keystore
```

Firma con la Llave de Mac OS X:

```
-alias AIRcert -storetype KeychainStore -providerName Apple
```

Firma con el almacén de claves del sistema de Windows:

```
-alias cn=AIRCert -storeype Windows-MY
```

Firma con un token de hardware (consulte las instrucciones del fabricante del token para ver cómo se configura Java para utilizar el token y para obtener el valor correcto de `providerName`):

```
-alias AIRCert -storetype pkcs11 -providerName tokenProviderName
```

Firma sin incorporar una marca de hora:

```
-storetype pkcs12 -keystore cert.p12 -tsa none
```

Empaquetado de una aplicación de AIR en un instalador nativo

En AIR 2, ADT puede utilizarse para crear instaladores de la aplicación nativos para distribuir aplicaciones de AIR. Por ejemplo, es posible crear un archivo de instalación EXE para la distribución de una aplicación de AIR en Windows. Se puede crear un archivo de instalación DMG para la distribución de una aplicación de AIR en Mac OS. Es posible crear un archivo de instalación DEB o RPM para la distribución de una aplicación de AIR en Linux.

Las aplicaciones instaladas con un instalador de aplicación nativo se conocen como aplicaciones de perfil de escritorio ampliadas. No es posible utilizar ADT para empaquetar un instalador nativo para una aplicación de AIR si el archivo descriptor de la aplicación no admite el perfil ampliado de escritorio. Este perfil se puede restringir utilizando el elemento `supportedProfiles` en el archivo descriptor de la aplicación. Consulte "Limitación de los perfiles de la aplicación de destino" en la página 77.

Se puede crear una versión del instalador nativo de la aplicación de AIR de dos formas básicas:

- Se puede crear el instalador nativo basado en el archivo descriptor de la aplicación y otros archivos de origen: (Otros archivos de origen pueden incluir archivos SWF, HTML y otros recursos.)
- El instalador nativo se puede crear en función de un archivo de AIR o un archivo AIRI.

ADT se debe utilizar en el mismo sistema operativo que el del archivo de instalación nativo que se desea generar. Por lo tanto, para crear un archivo EXE para Windows, ejecute ADT en Windows. Para crear un archivo DMG para Mac OS, ejecute ADT en Mac OS. Para crear un archivo DEB o RPG para Linux, ejecute ADT en Linux.

Cuando se crea un instalador nativo para distribuir una aplicación de AIR, la aplicación gana estas capacidades:

- Puede iniciar e interactuar con los procesos nativos, utilizando la clase NativeProcess. Para obtener más información, consulte una de las siguientes referencias:
  - Communicating with native processes in AIR (Comunicación con procesos nativos en AIR; en inglés.) (Para desarrolladores de HTML)

- Communicating with native processes in AIR (Comunicación con procesos nativos en AIR; en inglés) (Para desarrolladores HTML)

- Puede llamar al método `File.openWithDefaultApplication()` en cualquier archivo con la aplicación del sistema predeterminada definida para abrirlo, independientemente de su tipo de archivo. (Existen limitaciones en las aplicaciones que *no* están instaladas con un instalador nativo. Para obtener más información, consulte la entrada para `File.openWithDefaultApplication()` en la referencia del lenguaje.)

Si el usuario hace doble clic en el archivo de instalación nativo, se instalará la aplicación de AIR. Si la versión necesaria de Adobe AIR aún no está instalada en el equipo, el instalador la descarga de la red y la instala en primer lugar. Si no hay conexión de red con la que obtener la versión correcta de Adobe AIR (si es necesaria), se produce un error de instalación. Asimismo, la instalación falla si el sistema operativo no se admite en Adobe AIR 2.

Nota: *si desea que un archivo sea ejecutable en su aplicación instalada, asegúrese de que lo es en el sistema de archivos en el momento de empaquetar la aplicación. (En Mac y Linux, puede utilizar chmod para establecer el indicador de ejecutable, si es necesario.)*

Creación de un instalador nativo a partir de los archivos de origen de la aplicación

Para crear un instalador nativo a partir de los archivos de origen para la aplicación, utilice el comando `-package` con la siguiente sintaxis (en una sola línea de comandos):

```
adt -package AIR_SIGNING_OPTIONS -target native [WINDOWS_INSTALLER_SIGNING_OPTIONS]
installer_file app_xml [file_or_dir | -C dir file_or_dir | -e file dir ...] ...
```

Esta sintaxis es similar a la sintaxis para empaquetar un archivo de AIR (sin un instalador nativo). Sin embargo, existen algunas diferencias:

- La opción `-target native` se añade al comando. (Si se especifica `-target air`, ADT genera un archivo de AIR en lugar de un archivo de instalación nativo.)
- El archivo DMG o EXE de destino se especifica como `installer_file`.
- De forma opcional, en Windows es posible añadir un segundo conjunto de opciones de firma, indicado como `[WINDOWS_INSTALLER_SIGNING_OPTIONS]` en el listado de sintaxis. En Windows, además de firmar un archivo de AIR, se puede firmar el archivo de Windows Installer. Utilice el mismo tipo de certificado y sintaxis de opción de firma que se usaría para firmar el archivo de AIR (consulte "Opciones de firma en la línea de comandos de ADT" en la página 63). Se puede emplear el mismo certificado para firmar el archivo de AIR y el archivo de instalación, o bien, se pueden especificar certificados diferentes. Cuando un usuario descarga un archivo firmado de Windows Installer de la web, Windows identifica el origen del archivo, en función del certificado.

Para obtener más información sobre las opciones de ADT distintas a la opción `-target`, consulte "Empaquetado de archivos de instalación de AIR" en la página 54.

En el siguiente ejemplo se crea un archivo DMG (un archivo de instalación nativo para Mac OS):

```
adt -package -storetype pkcs12 -keystore myCert.pfx -target native myApp.dmg application.xml
index.html resources
```

En el siguiente ejemplo se crea un archivo EXE (un archivo de instalación nativo para Windows):

```
adt -package -storetype pkcs12 -keystore myCert.pfx -target native myApp.exe application.xml
index.html resources
```

En el siguiente ejemplo se crea un archivo EXE y se firma:

```
adt -package -storetype pkcs12 -keystore myCert.pfx -target native -storetype pkcs12 -keystore
myCert.pfx myApp.exe application.xml index.html resources
```

Creación de un instalador nativo a partir de un archivo de AIR o un archivo de AIRI

ADT se puede utilizar para generar un archivo de instalación nativo basado en un archivo de AIR o AIRI. Para crear un instalador nativo basado en un archivo de AIR, utilice el comando `-package` de ADT con la siguiente sintaxis (en una sola línea de comandos):

```
adt -package -target native [WINDOWS_INSTALLER_SIGNING_OPTIONS] installer_file air_file
```

Esta sintaxis es similar a la que se utiliza para crear un instalador nativo basado en los archivos de origen para la aplicación de AIR. Sin embargo, existen unas cuantas diferencias:

- Como origen, se especifica un archivo de AIR, en lugar de un archivo descriptor de la aplicación y otros archivos de origen para la aplicación de AIR.
- No especifique opciones de firma para el archivo de AIR, ya que ya está firmado.

Para crear un instalador nativo basado en un archivo de *AIRI*, utilice el comando `-package` de ADT con la siguiente sintaxis (en una sola línea de comandos):

```
adt AIR_SIGNING_OPTIONS -package -target native [WINDOWS_INSTALLER_SIGNING_OPTIONS]
installer_file airi_file
```

Esta sintaxis es similar a la que se emplea para crear un instalador nativo basado en un archivo de AIR. Sin embargo, existen unas cuantas diferencias:

- Como origen, se especifica un archivo de AIRI.
- Se especifican opciones de firma para la aplicación de AIR de destino.

En el siguiente ejemplo se crea un archivo DMG (un archivo de instalación nativo para Mac OS) basado en un archivo de AIR:

```
adt -package -target native myApp.dmg myApp.air
```

En el siguiente ejemplo se crea un archivo EXE (un archivo de instalación nativo para Windows) basado en un archivo de AIR:

```
adt -package -target native myApp.exe myApp.air
```

En el siguiente ejemplo se crea un archivo EXE (basado en un archivo de AIR) y se firma:

```
adt -package -target native -storetype pkcs12 -keystore myCert.pfx myApp.exe myApp.air
```

En el siguiente ejemplo se crea un archivo DMG (un archivo de instalación nativo para Mac OS) basado en un archivo de AIRI:

```
adt -storetype pkcs12 -keystore myCert.pfx -packag e-target native myApp.dmg myApp.airi
```

En el siguiente ejemplo se crea un archivo EXE (un archivo de instalación nativo para Windows) basado en un archivo de AIRI:

```
adt -storetype pkcs12 -keystore myCert.pfx -package -target native myApp.exe myApp.airi
```

En el siguiente ejemplo se crea un archivo EXE (basado en un archivo de AIRI) y se firma:

```
adt -package -storetype pkcs12 -keystore myCert.pfx -target native -storetype pkcs12 -keystore
myCert.pfx myApp.exe myApp.airi
```

Creación de archivos intermedios sin firmar de AIR con ADT

Utilice el comando `-prepare` para crear un archivo intermedio de AIR sin firmar. Para producir un archivo de instalación de AIR válido, el archivo intermedio de AIR debe firmarse con el comando `-sign` de ADT.

El comando `-prepare` acepta los mismos indicadores y parámetros que el comando `-package` (a excepción de las opciones de firma). La única diferencia es que no se firma el archivo de salida. El archivo intermedio se genera con la extensión del nombre de archivo `airi`.

Para firmar un archivo intermedio de AIR, utilice el comando `-sign` de ADT. (Consulte "Firma de un archivo intermedio de AIR con ADT" en la página 68).

Ejemplo del comando -prepare de ADT

```
adt -prepare unsignedMyApp.airi myApp.xml myApp.swf components.swc
```

Firma de un archivo intermedio de AIR con ADT

Para firmar un archivo intermedio de AIR con ADT, utilice el comando `-sign`. El comando "sign" sólo funciona con archivos intermedios de AIR (con la extensión `airi`). Un archivo de AIR no se puede firmar por segunda vez.

Para crear un archivo intermedio de AIR, utilice el comando de ADT `-sign`. (Consulte "Creación de archivos intermedios sin firmar de AIR con ADT" en la página 67.)

Firma de archivos AIRI

❖ Utilice el comando `-sign` de ADR con la sintaxis siguiente:

```
adt -sign SIGNING_OPTIONSairi_fileair_file
```

SIGNING_OPTIONS Las opciones de firma identifican la clave privada y el certificado con que se firma el archivo de AIR. Estas opciones se describen en "Opciones de firma en la línea de comandos de ADT" en la página 63.

airi_file La ruta al archivo intermedio sin firmar de AIR que se va a firmar.

air_file El nombre del archivo de AIR que se va a crear.

Ejemplo del comando -sign de ADT

```
adt -sign -storetype pkcs12 -keystore cert.p12 unsignedMyApp.airi myApp.air
```

Para obtener más información, consulte "Firma digital de archivos de AIR" en la página 96.

Firma de un archivo de AIR para cambiar el certificado de la aplicación

Para publicar una actualización para una aplicación de AIR existente mientras se utiliza un certificado de firma nuevo o renovado, emplee el comando `-migrate` de ADT para aplicar una firma de migración de certificados. Una firma de migración es una segunda firma aplicada a un archivo de AIR utilizando el certificado original. La firma de migración valida que una actualización de la aplicación fue generada por los propietarios del certificado original.

Para poder aplicar una firma de migración, el certificado original debe ser aún válido o haber caducado en los últimos 180 días. Una vez que el certificado haya caducado y haya transcurrido el período de gracia de 180 días, una firma de migración no se puede aplicar. Los usuarios de la aplicación tendrán que desinstalar la versión existente antes de poder instalar la versión actualizada. La firma de migración lleva una marca de hora, de forma predeterminada, de modo que las actualizaciones de AIR firmadas con firma de migración seguirán siendo válidas incluso una vez caducado el certificado.

Nota: *el período de gracia de 180 días sólo se aplica a las aplicaciones que especifican la versión de AIR 1.5.3, o superior, en el espacio de nombres del descriptor de la aplicación.*

Para migrar la aplicación y utilizar un nuevo certificado:

1 Cree una actualización de la aplicación.

2 Empaquete y firme el archivo de AIR de actualización con el certificado **nuevo**.

3 Vuelva a firmar el archivo de AIR con el certificado **original** utilizando el comando `-migrate`.

Un archivo de AIR firmado con el comando `-migrate` puede utilizarse tanto para instalar una nueva versión de la aplicación como para actualizar las versiones anteriores, incluidas las que fueron firmadas con el certificado anterior.

Nota: *al actualizar una aplicación publicada para una versión de AIR anterior a .5.3, es necesario especificar el ID de editor original en el descriptor de la aplicación. De lo contrario, los usuarios de la aplicación deben desinstalar la versión anterior antes de la instalar la actualización.*

Migración de una aplicación de AIR para utilizar un nuevo certificado

❖ Utilice el comando `-migrate` de ADT con la sintaxis siguiente:

```
adt -migrate SIGNING_OPTIONS air_file_in air_file_out
```

SIGNING_OPTIONS Las opciones de firma identifican la clave privada y el certificado con que se firma el archivo de AIR. Estas opciones deben identificar el certificado de firma **original** y se describen en "Opciones de firma en la línea de comandos de ADT" en la página 63

air_file_in El archivo de AIR para la actualización, firmado con el certificado **nuevo**.

air_file_out El archivo de AIR que se va a crear.

Ejemplo con ADT

```
adt –migrate -storetype pkcs12 -keystore cert.p12 myApp.air myApp.air
```

Para obtener más información, consulte "Firma digital de archivos de AIR" en la página 96.

Nota: *el comando* `-migrate` *se añadió a ADT en la versión 1.1 de AIR.*

Creación de certificados con firma automática con ADT

Los certificados con firma automática se pueden utilizar para generar un archivo de instalación de AIR válido. No obstante, estos certificados sólo proporcionan garantías de seguridad limitadas para los usuarios. La autenticidad de los certificados con firma automática no se puede verificar. Cuando se instala un archivo de AIR con firma automática, los datos del editor se presentan al usuario como "Desconocidos". Un certificado generado por ADT tiene una validez de cinco años.

Si se crea una actualización para una aplicación de AIR que se firmó con un certificado autogenerado, habrá que utilizar el mismo certificado para firmar tanto los archivos de AIR originales como los de la actualización. Los certificados que produce ADT son siempre únicos, aunque se utilicen los mismos parámetros. Por lo tanto, si desea firmar automáticamente las actualizaciones con un certificado generado por ADT, conserve el certificado original en un lugar seguro. Por otra parte, no podrá producir un archivo de AIR actualizado después de haber caducado el certificado generado por ADT. (Sí podrá publicar nuevas aplicaciones con otro certificado, pero no las nuevas versiones de la misma aplicación).

Importante: *dadas las limitaciones de los certificados con firma automática, Adobe recomienda enfáticamente utilizar un certificado comercial procedente de una entidad emisora de certificados de renombre para firmar aplicaciones de AIR que se distribuyen de forma pública.*

El certificado y la clave privada asociada generados por ADT se guardan en un archivo de almacén de claves del tipo PKCS12. La contraseña especificada se establece en la propia clave y no en el almacén de claves.

Generación de un certificado de ID digital para firmar automáticamente archivos de AIR

❖ Utilice el comando `-certificate` del ADT (en una sola línea de comandos):

```
adt -certificate -cn name [-ou org_unit][-o org_name][-c country]
key_type pfx_file password
```

-cn name La cadena asignada como nombre común del nuevo certificado.

-ou org_unit Una cadena asignada como unidad organizativa emisora del certificado. (Opcional.)

`-o org_name` Una cadena asignada como organización emisora del certificado. (Opcional.)

`-c country` Código de país de dos letras de la norma ISO-3166. Si el código suministrado no es válido, no se genera el certificado. (Opcional.)

`key_type` El tipo de clave que se va a utilizar para el certificado, que puede ser "1024-RSA" o "2048-RSA".

pfx_file La ruta del archivo del certificado que se va a generar.

password La contraseña para acceder al nuevo certificado. Cuando se firmen archivos de AIR con este certificado, se necesitará la contraseña.

Ejemplos de generación de certificados

```
adt -certificate -cn SelfSign -ou QE -o "Example, Co" -c US 2048-RSA newcert.p12 39#wnetx3tl
adt -certificate -cn ADigitalID 1024-RSA SigningCert.p12 39#wnetx3tl
```

Cuando se utilizan estos certificados para firmar archivos de AIR, hay que usar las siguientes opciones de firma con los comandos `-package` o `-prepare` de ADT:

```
-storetype pkcs12 -keystore newcert.p12 -keypass 39#wnetx3tl
-storetype pkcs12 -keystore SigningCert.p12 -keypass 39#wnetx3tl
```

Nota: *las versiones de Java 1.5 y posteriores no aceptan caracteres ASCII superior en contraseñas utilizadas para proteger archivos de certificado PKCS12. Utilice únicamente caracteres ASCII normales en la contraseña.*

Capítulo 11: Configuración de las propiedades de una aplicación de AIR

Aparte de todos los archivos y demás objetos que conforman una aplicación de AIR, cada aplicación de AIR requiere un archivo descriptor de la aplicación. El archivo descriptor de la aplicación es un archivo XML que define las propiedades básicas de la aplicación.

Muchos entornos de desarrollo que admiten AIR generan automáticamente un descripción de la aplicación cuando se crea un proyecto. De lo contrario, debe crear su propio archivo descriptor. Un archivo descriptor de ejemplo, `descriptor-sample.xml`, se puede encontrar en el directorio `samples` de los kits de desarrollo de software (SDK) de AIR y Flex.

Importante: no edite el archivo descriptor de la aplicación mientras esté abierto el cuadro de diálogo de Configuración de AIR de Flash Professional CS5. Guarde los cambios en el archivo descriptor de la aplicación antes de abrir el cuadro de diálogo de configuración de iPhone.

El archivo descriptor de la aplicación también se puede emplear para definir la configuración de una aplicación iPhone basada en ActionScript. Para obtener más información, consulte Configuración del descriptor de la aplicación para el desarrollo de iPhone.

Estructura del archivo descriptor de la aplicación

El archivo descriptor de la aplicación contiene propiedades que afectan al conjunto de la aplicación, como son el nombre, la versión, el aviso de copyright, etc. El archivo descriptor puede tener cualquier nombre de archivo. Al combinar la aplicación en un paquete, el nombre del archivo descriptor de la aplicación cambia a `application.xml` y el archivo se coloca en un directorio especial en el paquete de AIR.

El siguiente es un ejemplo de archivo descriptor de una aplicación:

```
<?xml version="1.0" encoding="utf-8" ?>
<application xmlns="http://ns.adobe.com/air/application/2.0">
    <id>HelloWorld</id>
    <version>2.0</version>
    <filename>Hello World</filename>
    <name>Example Co. AIR Hello World</name>
     <description>
        <text xml:lang="en">This is a example.</text>
        <text xml:lang="fr">C'est un exemple.</text>
        <text xml:lang="es">Esto es un ejemplo.</text>
    </description>
    <copyright>Copyright (c) 2006 Example Co.</copyright>
    <initialWindow>
        <title>Hello World</title>
        <content>
            HelloWorld.swf
        </content>
        <systemChrome>none</systemChrome>
        <transparent>true</transparent>
        <visible>true</visible>
        <minimizable>true</minimizable>
```

```
        <maximizable>false</maximizable>
        <resizable>false</resizable>
        <width>640</width>
        <height>480</height>
        <minSize>320 240</minSize>
        <maxSize>1280 960</maxSize>
    </initialWindow>
    <installFolder>Example Co/Hello World</installFolder>
    <programMenuFolder>Example Co</programMenuFolder>
    <icon>
        <image16x16>icons/smallIcon.png</image16x16>
        <image32x32>icons/mediumIcon.png</image32x32>
        <image48x48>icons/bigIcon.png</image48x48>
        <image128x128>icons/biggestIcon.png</image128x128>
    </icon>
    <customUpdateUI>true</customUpdateUI>
    <allowBrowserInvocation>false</allowBrowserInvocation>
    <fileTypes>
        <fileType>
            <name>adobe.VideoFile</name>
            <extension>avf</extension>
            <description>Adobe Video File</description>
            <contentType>application/vnd.adobe.video-file</contentType>
            <icon>
                <image16x16>icons/avfIcon_16.png</image16x16>
                <image32x32>icons/avfIcon_32.png</image32x32>
                <image48x48>icons/avfIcon_48.png</image48x48>
                <image128x128>icons/avfIcon_128.png</image128x128>
            </icon>
        </fileType>
    </fileTypes>
</application>
```

Si la aplicación utiliza un archivo HTML como su contenido raíz en lugar de un archivo SWF, únicamente el elemento `<content>` es diferente:

```
<content>
    HelloWorld.html
</content>
```

Definición de propiedades en el archivo descriptor de la aplicación

Utilice los atributos y elementos XML del descriptor de la aplicación para definir los siguientes tipos de propiedades para su aplicación de AIR:

- Versión necesaria del motor de ejecución de AIR.
- Identidad de la aplicación.
- Carpetas de instalación y de menús de programa.
- Propiedades de ventana y contenido inicial.
- Archivos de icono de la aplicación.

- Si la aplicación proporciona una interfaz de usuario personalizada para actualizaciones.
- Si el contenido SWF que se ejecuta en el navegador del usuario puede invocar la aplicación.
- Asociaciones de tipos de archivos.

Especificación de la versión necesaria de AIR

Los atributos del elemento raíz de un archivo descriptor de la aplicación, `application`, especifican la versión necesaria del motor de ejecución de AIR:

```
<application xmlns="http://ns.adobe.com/air/application/2.0">
```

xmlns El espacio de nombre de AIR, el cual debe definirse como espacio de nombre XML predeterminado. El espacio de nombre cambia con cada edición principal de AIR (pero no con las revisiones menores). El último segmento del espacio de nombre, por ejemplo "2.0", indica la versión del motor de ejecución que requiere la aplicación. Asegúrese de establecer el espacio de nombres en AIR 2 (`"http://ns.adobe.com/air/application/2.0"`) si su aplicación utiliza alguna nueva función de AIR 2.

Para las aplicaciones basadas en SWF, la versión del motor de ejecución de AIR especificada en el descriptor de la aplicación determina la versión de SWF máxima que se puede cargar como contenido inicial de la aplicación. Las aplicaciones que especifican AIR 1.0 ó 1.1 sólo pueden utilizar archivos SWF9 (Flash Player 9) como contenido inicial, aunque se ejecuten utilizando el motor de ejecución de AIR 2. Las aplicaciones que especifican AIR 1.5 (o posterior) pueden usar archivos SWF9 o SWF10 (Flash Player 10) como contenido inicial.

La versión de SWF determina qué versión de las API de AIR y Flash Player están disponibles. Si un archivo SWF9 se utiliza como contenido inicial de una aplicación de AIR 1.5, esa aplicación sólo tendrá acceso a las API de AIR 1.1 y Flash Player 9. Asimismo, no se aplicarán los cambios de comportamiento realizados en las API existentes en AIR 2.0 o Flash Player 10.1. (Los cambios importantes relacionados con la seguridad en las API constituyen una excepción en este principio y se pueden aplicar de forma retroactiva en revisiones actuales o posteriores del motor de ejecución.)

Para las aplicaciones basadas en HTML, la versión del motor de ejecución especificada en el descriptor de la aplicación determina qué versión de las API de AIR y Flash Player están disponibles en la aplicación. Los comportamientos de HTML, CSS y JavaScript siempre están determinados por la versión del Webkit utilizado en el motor de ejecución de AIR instalado, y no por el descriptor de la aplicación.

Si una aplicación de AIR carga contenido SWF, la versión de las API de AIR y Flash Player disponible para dicho contenido depende del modo en que se cargue el contenido. En ocasiones la versión eficaz se determina mediante el espacio de nombres del descriptor de la aplicación, otras veces mediante la versión del contenido de carga y a veces mediante la versión del contenido cargado. La siguiente tabla muestra cómo la versión de la API está determinada según el método de carga:

Modo de carga del contenido	**Forma en que la versión de la API se ve determinada**
Contenido inicial, aplicación basada en SWF	Versión SWF del archivo **cargado**
Contenido inicial, aplicación basada en HTML	Espacio de nombres del descriptor de la aplicación
SWF cargado mediante contenido SWF	Versión del contenido de **carga**

Modo de carga del contenido	Forma en que la versión de la API se ve determinada
Biblioteca SWF cargada mediante contenido HTML utilizando la etiqueta <script>	Espacio de nombres del descriptor de la aplicación
SWF cargado mediante contenido HTML utilizando las API de AIR o Flash Player (por ejemplo, flash.display.Loader)	Espacio de nombres del descriptor de la aplicación
SWF cargado mediante contenido HTML utilizando etiquetas <object> o <embed> (o las API equivalentes de JavaScript)	Versión SWF del archivo **cargado**

Al cargar un archivo SWF de una versión distinta al contenido de carga, puede encontrarse con los siguientes problemas:

- Carga de contenido SWF10 mediante SWF9 (o anterior): las referencias a las API de AIR 1.5 y Flash Player 10 en el contenido cargado no se resolverán.
- Carga de contenido SWF9 (o anterior) mediante SWF10: las API modificadas en AIR 1.5 y Flash Player 10 pueden comportarse de un modo inesperado para el contenido cargado.

Definición de la identidad de la aplicación

Los elementos siguientes definen el ID de la aplicación, el ID de editor, la versión, el nombre, el nombre de archivo, la descripción y el aviso de copyright de la aplicación:

```
<id>TestApp</id>
<publisherID>48B5E02D9FB21E6389F73B8D17023320485DF8CE.1</publisherID>
<version>2.0</version>
<filename>TestApp</filename>
<name>
    <text xml:lang="en">Hello AIR</text>
    <text xml:lang="fr">Bonjour AIR</text>
    <text xml:lang="es">Hola AIR</text>
</name>
<description>An MP3 player.</description>
<copyright>Copyright (c) 2008 YourCompany, Inc.</copyright>
```

id Cadena de identificación de la aplicación, denominada ID de la aplicación. El valor del atributo se limita a los siguientes caracteres:

- 0–9
- a–z
- A–Z
- . (punto)
- - (guión)

El valor debe contener entre 1 y 212 caracteres. Este elemento es obligatorio.

publisherID (Opcional) Se utiliza al actualizar una aplicación de AIR de la versión 1.5.2 o anterior a AIR 1.5.3 o posterior. Este elemento contiene el ID de editor de la versión anterior de la aplicación de AIR. Especifique este elemento sólo si está desarrollando una aplicación para AIR 1.5.3 o posterior y existe una versión de la aplicación que utilice AIR 1.5.2 o anterior. Para obtener más información, consulte “Identificador del editor de AIR” en la página 98.

Si está desarrollando una aplicación de AIR para AIR 1.5.3 o posterior y existe una versión que utilice AIR 1.5.2 o anterior:

1 Determine el ID de editor actual de la aplicación. En una aplicación instalada, el ID de editor se encuentra en el archivo META-INF/AIR/publisherid.

2 Añada el ID de editor al elemento `<publisherID>` del archivo descriptor de la aplicación para su nueva aplicación.

version Especifica la información sobre la versión para la aplicación. (No tiene nada que ver con la versión del motor de ejecución). La cadena de la versión es un designador definido por la aplicación. AIR no interpreta la cadena de versión de ningún modo. Por ejemplo, no supone que la versión "3.0" es más actualizada que la versión "2.0". Ejemplos: `"1.0"`, `".4"`, `"0.5"`, `"4.9"`, `"1.3.4a"`. Este elemento es obligatorio.

filename La cadena que se deberá utilizar como nombre de archivo de la aplicación (sin la extensión) al instalar ésta. El archivo de la aplicación inicia la aplicación de AIR en el motor de ejecución. Si no se facilita ningún valor para `name`, el nombre de archivo (valor de `filename`) se utilizará también como nombre de la carpeta de instalación. Este elemento es obligatorio.

La propiedad `filename` puede contener cualquier carácter Unicode (UTF-8) salvo los siguientes, cuyo uso en nombres de archivo está prohibido en diversos sistemas de archivos:

Carácter	Código hexadecimal
diversos	0x00 – x1F
*	x2A
"	x22
:	x3A
>	x3C
<	x3E
?	x3F
\	x5C
|	x7C

El valor de `filename` no puede terminar con un punto.

name (opcional, pero recomendado) El título que aparece en el instalador de aplicaciones de AIR.

Si especifica un solo nodo de texto (en lugar de varios elementos `text`), el instalador de aplicaciones de AIR utiliza este nombre, cualquiera que sea el idioma seleccionado para el sistema:

```
<name>Test Application</name>
```

El esquema del descriptor de aplicaciones de AIR 1.0 sólo permite definir un nodo de texto para el nombre (y no varios elementos de `text`).

En AIR 1.1 (o posterior) se pueden especificar varios idiomas en el elemento `name`. En el ejemplo siguiente se especifica el nombre en tres idiomas (inglés, francés y español):

```
<name>
    <text xml:lang="en">Hello AIR</text>
    <text xml:lang="fr">Bonjour AIR</text>
    <text xml:lang="es">Hola AIR</text>
</name>
```

El atributo `xml:lang` para cada elemento de texto especifica un código de idioma, de conformidad con lo definido en RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

El instalador de aplicaciones de AIR utiliza el nombre que mejor se corresponde con el idioma de la interfaz de usuario del sistema operativo del usuario. Por ejemplo, tomemos una instalación en la que el elemento `name` del archivo descriptor de la aplicación incluye un valor para la configuración regional "es" (español). El instalador de aplicaciones de AIR utiliza el nombre "es" si el sistema operativo identifica "es" (español) como el idioma de la interfaz de usuario. También utiliza el nombre "es" si el idioma de la interfaz de usuario del sistema es es-ES (español de España). Sin embargo, si el idioma de la interfaz de usuario es es-ES y el archivo descriptor de la aplicación define tanto el nombre es-ES como el nombre es-UY, el instalador de aplicaciones de AIR utiliza el valor es-ES. Si la aplicación no define ningún nombre que coincide con el idioma de la interfaz de usuario del sistema, el instalador de aplicaciones de AIR utiliza el primer valor de `name` que se define en el archivo descriptor de la aplicación.

Si no se especifica ningún elemento `name`, el instalador de aplicaciones de AIR muestra el nombre de archivo definido para `filename` como el nombre de la aplicación.

El elemento `name` sólo define el título de la aplicación que se utiliza en el instalador de aplicaciones de AIR. El instalador de aplicaciones de AIR admite varios idiomas: chino tradicional, chino simplificado, checo, neerlandés, inglés, francés, alemán, italiano, japonés, coreano, polaco, portugués brasileño, ruso, español, sueco y turco. El instalador de aplicaciones de AIR selecciona el idioma visualizado (para texto que no sea el título y la descripción de la aplicación) con base en el idioma de la interfaz de usuario del sistema. Esta selección de idioma es independiente de la configuración del archivo descriptor de la aplicación.

El elemento `name`*no* define las configuraciones regionales disponibles para la aplicación instalada y en funcionamiento. Para obtener más información sobre el desarrollo de aplicaciones multilingües, consulte “Localización de aplicaciones de AIR” en la página 142.

description (opcional) La descripción de la aplicación. Se muestra en el instalador de aplicaciones de AIR.

Si especifica un solo nodo de texto (en lugar de varios elementos para text), el instalador de aplicaciones de AIR utiliza este nombre, cualquiera que sea el idioma seleccionado para el sistema:

```
<description>This is a sample AIR application.</description>
```

El esquema del descriptor de aplicaciones de AIR 1.0 sólo permite definir un nodo de texto para el nombre (y no varios elementos de `text`).

En AIR 1.1 (o superior) se pueden especificar varios idiomas en el elemento `description`. En el ejemplo siguiente se especifica una descripción en tres idiomas (inglés, francés y español):

```
<description>
    <text xml:lang="en">This is a example.</text>
    <text xml:lang="fr">C'est un exemple.</text>
    <text xml:lang="es">Esto es un ejemplo.</text>
</description>
```

El atributo `xml:lang` para cada elemento de texto especifica un código de idioma, de conformidad con lo definido en RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

El instalador de aplicaciones de AIR utiliza la descripción que es más similar al idioma de la interfaz de usuario del sistema operativo del usuario. Por ejemplo, tomemos una instalación en la que el elemento `description` del archivo descriptor de la aplicación incluye un valor para la configuración regional "es" (española). El instalador de aplicaciones de AIR utiliza el nombre "es" si el sistema del usuario identifica "es" (español) como el idioma de la interfaz de usuario. También utiliza el nombre "es" si el idioma de la interfaz de usuario del sistema es es-ES (español de España). Sin embargo, si el idioma de la interfaz de usuario del sistema es es-ES y el archivo descriptor de la aplicación define tanto el nombre es-ES como el nombre es-UY, el instalador de aplicaciones de AIR utiliza el valor es-ES. Si la aplicación no define ningún nombre que coincide con el idioma de la interfaz de usuario del sistema, el instalador de aplicaciones de AIR utiliza el primer valor de `description` que se define en el archivo descriptor de la aplicación.

Para obtener más información sobre el desarrollo de aplicaciones multilingües, consulte "Localización de aplicaciones de AIR" en la página 142.

copyright (opcional) La información de copyright para la aplicación de AIR. En Mac OS el aviso de copyright aparece en el cuadro de diálogo Acerca de para la aplicación instalada. En Mac OS, la información de copyright también se utiliza en el campo NSHumanReadableCopyright del archivo Info.plist para la aplicación.

Limitación de los perfiles de la aplicación de destino

Con las aplicaciones de iPhone y AIR 2 creadas con ActionScript 3.0, es posible limitar el perfil de destino de la aplicación compilada.

```
<supportedProfiles>extendedDesktop</supportedProfiles>
```

supportedProfiles: (opcional) identifica los perfiles compatibles con la aplicación.

Se pueden incluir cualquiera de estos tres valores en el elemento supportedProfiles:

- `desktop`: el perfil de escritorio define las aplicaciones de AIR que están instaladas en un equipo de escritorio utilizando un archivo de AIR. Estas aplicaciones no tienen acceso a la clase NativeProcess (que proporciona comunicación con las aplicaciones nativas).
- `extendedDesktop`: el perfil de escritorio ampliado define las aplicaciones de AIR que se encuentran instaladas en un equipo de escritorio utilizando un instalador de aplicaciones nativas. Estas aplicaciones tienen acceso a la clase NativeProcess (que proporciona comunicación con las aplicaciones nativas). Para obtener más información, consulte "Empaquetado de una aplicación de AIR en un instalador nativo" en la página 65. Asimismo, consulte Communicating with native processes in AIR (Comunicación con procesos nativos en AIR; en inglés) (para desarrolladores de ActionScript) y Communicating with native processes in AIR (Comunicación con procesos nativos en AIR; en inglés) (para desarrolladores HTML).
- `mobileDevice`: el perfil de dispositivo móvil define las aplicaciones de iPhone creadas con ActionScript 3.0.
- `extendedMobileDevice`: el perfil de dispositivo móvil ampliado no está en uso actualmente.

La propiedad `supportedProfiles` es opcional. Si no se incluye este elemento en el archivo descriptor de la aplicación, ésta se puede compilar e implementar para cualquier perfil.

Para especificar varios perfiles, separe cada uno de ellos con carácter de espacio. Por ejemplo, la siguiente configuración especifica que la aplicación sólo está disponible en los perfiles ampliados y el escritorio:

```
<supportedProfiles>desktop extendedDesktop</supportedProfiles>
```

Para obtener más información, consulte "Perfiles de la aplicación" en la página 83.

Definición de la carpeta de instalación y la carpeta de menús de programa

Las carpetas de instalación y de menús de programa se definen con la siguiente configuración de propiedades:

```
<installFolder>Acme</installFolder>
<programMenuFolder>Acme/Applications</programMenuFolder>
```

installFolder (opcional) Identifica el subdirectorio del directorio de instalación predeterminado.

En Windows el subdirectorio de instalación predeterminado es el directorio Archivos de programa. En Mac OS es el directorio /Aplicaciones. En Linux, es /opt/. Por ejemplo, si la propiedad `installFolder` está definida en `"Acme"` y una aplicación lleva el nombre `"EjemploApl"`, la aplicación se instala en C:\Archivos de programa\Acme\ExampleApp en Windows, en /Aplicaciones/Acme/Example.app en MacOS y /opt/Acme/ExampleApp en Linux.

Utilice la barra diagonal (/) como carácter separador entre directorios si desea especificar un subdirectorio anidado, como en el ejemplo siguiente:

```
<installFolder>Acme/Power Tools</installFolder>
```

La propiedad `installFolder` puede contener cualquier carácter Unicode (UTF-8) excepto aquellos cuyo uso en nombres de archivo está prohibido en diversos sistemas de archivos (para ver la lista de excepciones, consulte la anterior propiedad `filename`).

La propiedad `installFolder` es opcional. Si no se especifica ninguna propiedad para `installFolder`, la aplicación se instala en un subdirectorio del directorio de instalación predeterminado basado en la propiedad `name`.

programMenuFolder (Opcional) Identifica dónde deben guardarse los accesos directos a la aplicación en el menú Todos los programas del sistema operativo Windows o en el menú de aplicaciones de Linux. (En otros sistemas operativos, en la actualidad se hace caso omiso a esta opción). Las restricciones sobre el número de caracteres que se admiten en el valor de la propiedad son las mismas que para la propiedad `installFolder`. *No* utilice una barra diagonal (/) como último carácter de este valor.

Definición de las propiedades de la ventana inicial de la aplicación

Cuando se carga una aplicación de AIR, el motor de ejecución utiliza los valores del elemento `initialWindow` para crear la ventana inicial de la aplicación. Luego el motor de ejecución carga en la ventana el archivo SWF o HTML especificado en el elemento `content`.

El siguiente es un ejemplo del elemento `initialWindow`:

```
<initialWindow>
    <content>AIRTunes.swf</content>
    <title>AIR Tunes</title>
    <systemChrome>none</systemChrome>
    <transparent>true</transparent>
    <visible>true</visible>
    <minimizable>true</minimizable>
    <maximizable>true</maximizable>
    <resizable>true</resizable>
    <width>400</width>
    <height>600</height>
    <x>150</x>
    <y>150</y>
    <minSize>300 300</minSize>
    <maxSize>800 800</maxSize>
</initialWindow>
```

Si el contenido raíz de la aplicación es un archivo HTML en lugar de un archivo SWF, el elemento `content` simplemente hace referencia al archivo HTML:

```
<content>AIRTunes.html</content>
```

Los elementos secundarios del elemento `initialWindow` definen las propiedades de la ventana en la que se carga el archivo de contenido raíz.

content El valor especificado para el elemento `content` es la URL para el archivo principal de contenido de la aplicación. Éste será un archivo SWF o un archivo HTML. La URL se especifica relativa a la raíz de la carpeta de instalación de la aplicación. (Al ejecutar una aplicación de AIR con ADL, la URL se indica relativa a la carpeta que contiene el archivo descriptor de la aplicación. Se puede utilizar el parámetro `root-dir` de ADL para especificar otro directorio raíz).

Nota: *al tratarse como una URL el valor del elemento content, los caracteres del nombre del archivo de contenido deben codificarse como URL de acuerdo con las reglas definidas en RFC 1738. Los caracteres de espacio, por ejemplo, deben codificarse como* `%20`.

title (Opcional) El título de la ventana.

systemChrome (Opcional) Si este atributo se define en `standard`, se mostrará el fondo cromático estándar proporcionado por el sistema operativo. Si lo define en `none`, no se mostrará ningún fondo cromático del sistema.

Al utilizar el componente mx:WindowedApplication de Flex, si el fondo cromático del sistema no está configurado, el componente aplica su fondo cromático personalizado. La opción de fondo cromático no puede modificarse durante el tiempo de ejecución.

transparent (opcional) Defínalo en `"true"` si desea que la ventana de la aplicación admita la mezcla alfa. Una ventana con transparencia puede dibujarse más lentamente y necesitar más memoria. La opción de transparencia no puede modificarse durante el tiempo de ejecución.

Importante: *sólo se puede establecer* `transparent` *en* `true` *si* `systemChrome` *se establece en* `none`.

visible (opcional) El valor predeterminado es `false`. Defínalo en `true` solamente si desea que la ventana principal quede visible en cuanto se haya creado, lo cual también mostrará los cambios de la ventana a medida que se colocan sus componentes.

El componente mx:WindowedApplication de Flex muestra y activa la ventana de forma automática inmediatamente antes de distribuir el evento `applicationComplete`, a menos que el atributo `visible` esté definido en `false` en la definición MXML.

Puede convenir dejar la ventana principal oculta al principio para que no se muestren los ajustes de la posición y el tamaño de la ventana y la disposición del contenido. Se podrá mostrar después la ventana llamando al método `activate()` de la ventana o cambiando la propiedad `visible` a `true`.

x, y, width, height (opcionales) Límites iniciales de la ventana principal de la aplicación. Si no se definen estos valores, el tamaño de la ventana quedará definido por las opciones del archivo SWF raíz o, en el caso de HTML, por el sistema operativo. El valor máximo para `width` y `height` es de 2880 cada uno.

minSize, maxSize (opcionales) Tamaños mínimo y máximo de la ventana. Si no se definen estos valores, los determinará el sistema operativo.

minimizable, maximizable, resizable (opcionales) Especifican si se puede minimizar, maximizar y redimensionar la ventana. Estas opciones tienen el valor predeterminado `true`.

Nota: *en los sistemas operativos como Mac OS X en los cuales la maximización de las ventanas es una operación de redimensionamiento, para que la ventana no cambie de tamaño, tanto "maximizable" como "resizable" deben definirse en* `false`.

Más temas de ayuda

Incorporación de contenido SWF en HTML (para desarrolladores de ActionScript)

Incorporación de contenido SWF en HTML (para desarrolladores de HTML)

Adición de contenido PDF en AIR (para desarrolladores de ActionScript)

Adición de contenido PDF en AIR (para desarrolladores de HTML)

Trabajo con ventanas nativas de AIR (para desarrolladores de ActionScript)

Trabajo con ventanas nativas de AIR (para desarrolladores de HTML)

Especificación de archivos de iconos

La propiedad `icon` especifica un archivo de icono (o varios) a utilizar para la aplicación. Los iconos son opcionales. Si no se define la propiedad `icon`, el sistema operativo muestra un icono predeterminado.

La ruta se indica relativa al directorio raíz de la aplicación. Los archivos de iconos deben tener el formato PNG. Se pueden especificar todos los tamaños de icono siguientes:

```
<icon>
    <image16x16>icons/smallIcon.png</image16x16>
    <image32x32>icons/mediumIcon.png</image32x32>
    <image48x48>icons/bigIcon.png</image48x48>
    <image128x128>icons/biggestIcon.png</image128x128>
</icon>
```

Si hay un elemento para un tamaño determinado, la imagen que contiene el archivo debe ser exactamente del tamaño especificado. Si no se proporcionan todos los tamaños, el sistema operativo ajusta el tamaño más parecido para un uso determinado del icono.

Nota: *los iconos especificados no se añaden automáticamente al paquete de AIR. Los archivos de iconos deben estar incluidos en sus lugares relativos correctos cuando se empaqueta de la aplicación.*

Para obtener el mejor resultado posible, proporcione una imagen para cada uno de los tamaños disponibles. Asegúrese también de que los iconos sean de buen aspecto tanto en el modo de color de 16 bits como en el de 32 bits.

Interfaz de usuario personalizada para actualizaciones

AIR instala y actualiza las aplicaciones utilizando los cuadros de diálogo de instalación predeterminados. No obstante, puede proporcionar su propia interfaz para actualizar una aplicación. Para indicar que la aplicación debe encargarse del proceso de actualización, defina el elemento `customUpdateUI` en `true`:

```
<customUpdateUI>true</customUpdateUI>
```

Cuando la versión instalada de la aplicación tiene el elemento `customUpdateUI` definido como `true` y el usuario hace doble clic en el archivo de AIR para una nueva versión o instala una actualización de la aplicación utilizando la función de instalación integrada, el motor de ejecución abre la versión instalada de la aplicación. El motor de ejecución no abre el instalador de aplicaciones de AIR predeterminado. La lógica de la aplicación determina entonces cómo continuar con la operación de actualización. (Para que se lleve a cabo una actualización, el ID de la aplicación y del editor que figuran en el archivo de AIR deben coincidir con los valores de la aplicación instalada).

Nota: *el mecanismo* `customUpdateUI` *sólo entra en juego si la aplicación ya está instalada y el usuario hace doble clic en el archivo de instalación de AIR que contiene una actualización, o si instala una actualización de la aplicación utilizando la función de instalación integrada. Puede descargar una actualización e iniciarla a través de la lógica de su propia aplicación, visualizando la interfaz de usuario personalizada según convenga, esté o no* `customUpdateUI` *definido en* `true`*.*

Para obtener más información, consulte "Actualización de aplicaciones de AIR" en la página 105.

Inicio de la aplicación desde el navegador

Si se especifica la siguiente opción, se podrá iniciar la aplicación de AIR instalada utilizando la función de invocación desde el navegador (el usuario selecciona un vínculo en una página en el navegador web):

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

El valor predeterminado es `false`.

Si este valor se define en `true`, asegúrese de tener en cuenta las posibles consecuencias para la seguridad. Estos aspectos se describen en Invocación de una aplicación de AIR desde el navegador (para desarrolladores de ActionScript) e Invocación de una aplicación de AIR desde el navegado (para desarrolladores de HTML).

Para obtener más información, consulte "Instalación y ejecución de aplicaciones de AIR desde una página web" en la página 87.

Declaración de asociaciones con tipos de archivos

El elemento `fileTypes` permite declarar los tipos de archivos con que se puede asociar una aplicación de AIR. Cuando se instala una aplicación de AIR, los tipos de archivos declarados se registran en el sistema operativo y, si estos tipos de archivos aún no se encuentran asociados con ninguna otra aplicación, se asocian con la aplicación de AIR. Para suprimir una asociación existente entre un tipo de archivo y otra aplicación, utilice el método en tiempo de ejecución `NativeApplication.setAsDefaultApplication()` (preferentemente con el permiso del usuario).

Nota: *los métodos del motor de ejecución sólo pueden manejar asociaciones para los tipos de archivos declarados en el descriptor de la aplicación.*

```
<fileTypes>
    <fileType>
        <name>adobe.VideoFile</name>
        <extension>avf</extension>
        <description>Adobe Video File</description>
        <contentType>application/vnd.adobe.video-file</contentType>
        <icon>
            <image16x16>icons/AIRApp_16.png</image16x16>
            <image32x32>icons/AIRApp_32.png</image32x32>
            <image48x48>icons/AIRApp_48.png</image48x48>
            <image128x128>icons/AIRApp_128.png</image128x128>
        </icon>
    </fileType>
</fileTypes>
```

El elemento `fileTypes` es opcional. Puede contener varios elementos `fileType`.

Los elementos `name` y `extension` son obligatorios para cada declaración de `fileType` que se incluya. Se puede utilizar el mismo nombre con varias extensiones. La extensión identifica el tipo de archivo. (Especifique la extensión sin el punto anterior.) El elemento `description` es opcional; lo presenta al usuario la interfaz de usuario del sistema operativo. `contentType` es necesario en AIR 1.5 (era opcional en AIR 1.0 y 1.1). La propiedad ayuda al sistema operativo a localizar la mejor aplicación para abrir un archivo en determinadas circunstancias. El valor debe ser el tipo MIME del contenido del archivo. Se debe tener en cuenta que el valor se omite en Linux si el tipo de archivo ya se ha registrado y dispone de un tipo MIME asignado.

Se pueden especificar iconos para la extensión de archivo utilizando el mismo formato que el elemento "icon" de la aplicación. Los archivos de iconos también deben incluirse en el archivo de instalación de AIR (no se colocan automáticamente en el paquete).

Si hay un tipo de archivo asociado con una aplicación de AIR, cada vez que un usuario abra un archivo de ese tipo se invocará la aplicación. Si la aplicación ya está funcionando, AIR distribuye el objeto InvokeEvent a la instancia que se esté ejecutando. En caso contrario, AIR iniciará primero la aplicación. En ambos casos la ruta al archivo puede recuperarse del objeto InvokeEvent distribuido por el objeto NativeApplication. Puede utilizarse esta ruta para abrir el archivo.

Para obtener más información, consulte las siguientes secciones:

- Gestión de asociaciones con archivos (para desarrolladores de ActionScript)
- Gestión de asociaciones con archivos (para desarrolladores de HTML)
- Captura de argumentos de la línea de comandos {para desarrolladores de ActionScript)
- Captura de argumentos de la línea de comandos (para desarrolladores de HTML)

Capítulo 12: Perfiles de la aplicación

Adobe AIR 2 introduce el concepto de *perfiles*. Un perfil se define como un conjunto de capacidades y APIs admitidas. Cada perfil se puede instalar en un conjunto definido de equipos o dispositivos.

Por ejemplo, tanto el perfil de escritorio como el perfil ampliado se utilizan para las aplicaciones de escritorio de AIR. Sin embargo, una aplicación de perfil de escritorio ampliado se puede comunicar con procesos nativos. (Una aplicación de perfil de escritorio no puede.) Del mismo modo, el perfil móvil define un conjunto de capacidades que difieren de las disponibles en otros perfiles.

Existen cuatro perfiles:

Escritorio El perfil de escritorio define un conjunto de capacidades para aplicaciones de AIR que se instalan como archivos de AIR en un equipo de escritorio. Estas aplicaciones se instalarán y ejecutarán en plataformas de escritorio compatibles (Mac OS, Windows y Linux). Las aplicaciones de AIR desarrolladas en versiones de AIR anteriores a AIR 2, se pueden considerar dentro del perfil de escritorio. Algunas APIs no funcionan en este perfil. Por ejemplo, las aplicaciones de escritorio no se pueden comunicar con procesos nativos.

Escritorio ampliado El perfil de escritorio ampliado define un conjunto de capacidades para aplicaciones de AIR que están empaquetadas e instaladas con un instalador nativo. Estos instaladores nativos son archivos EXE en Windows, archivos DMG en Mac OS y archivos DEB o RPM en Linux. Las aplicaciones de escritorio ampliadas cuentan con capacidades adicionales que no están disponibles en las aplicaciones de perfil de escritorio. El empaquetado e implementación de aplicaciones de AIR con un instalador nativo es una nueva función de AIR 2. Para obtener más información, consulte “Empaquetado de una aplicación de AIR en un instalador nativo” en la página 65.

Dispositivo móvil El perfil de dispositivo móvil define un conjunto si las capacidades para las aplicaciones están instaladas en dispositivos móviles. ActionScript 3.0 y las APIs de AIR se pueden utilizar para crear aplicaciones para iPhone, iPod táctil y iPad. Actualmente, estos son los únicos dispositivos que admiten aplicaciones de perfil de dispositivo móvil.

Dispositivo móvil ampliado El perfil de dispositivo móvil ampliado define un conjunto si las capacidades para las aplicaciones se instalan en un subconjunto de dispositivos móviles. Este subconjunto de dispositivos móviles puede utilizar la clase HTMLLoader además de la funcionalidad definida para el perfil de dispositivo móvil. Actualmente, no existen dispositivos que admitan este perfil.

Restricción de perfiles de destino en el archivo descriptor de la aplicación

En AIR 2, el archivo descriptor de la aplicación incluye un elemento `supportedProfiles`, que permite restringir perfiles de destino. Por ejemplo, la siguiente configuración especifica que la aplicación sólo está disponible en el perfil de escritorio:

```
<supportedProfiles>desktop</supportedProfiles>
```

Cuando se establece este elemento, la aplicación sólo se puede empaquetar en los perfiles especificados. Utilice los siguientes valores:

- `desktop`: perfil de escritorio.
- `extendedDesktop`: perfil de escritorio ampliado.
- `mobileDevice`: perfil de dispositivo móvil.

El elemento `supportedProfiles` es opcional. Si no se incluye este elemento en el archivo descriptor de la aplicación, ésta se puede empaquetar e implementar para cualquier perfil.

Para especificar varios perfiles en el elemento `supportedProfiles`, separe cada uno de ellos con un carácter de espacio, tal y como se indica a continuación:

```
<supportedProfiles>desktop extendedDesktop</supportedProfiles>
```

Capacidades en diferentes perfiles

Algunas APIs de ActionScript no funcionan en determinados perfiles.

Escritorio

Las aplicaciones de perfil de escritorio tienen acceso a todas las APIs de ActionScript, con las siguientes excepciones:

API	Prueba de compatibilidad	Comentarios
Accelerometer	`Accelerometer.isSupported`	La compatibilidad con el acelerómetro sólo está disponible en el perfil del dispositivo móvil.
CameraRoll	`CameraRoll.isSupported`	Esta clase proporciona acceso a la carpeta Camera Roll de un dispositivo iPhone.
`File.openWithDefaultApplication()`	—	Las aplicaciones de perfil de escritorio pueden llamar a este método. Sin embargo, las aplicaciones de perfil de escritorio no pueden abrir todos los tipos de archivos. La descripción en la Referencia del lenguaje ActionScript 3.0 incluye los tipos de archivo que están restringidos.
`File.cacheAsBitmapMatrix`	—	Esta propiedad sólo se admite en las aplicaciones de ActionScript 3.0 en iPhone.
Geolocation	`Geolocation.isSupported`	La compatibilidad con la localización geográfica sólo está disponible en el perfil de dispositivo móvil.
`NativeApplication.systemIdleMode`	Ninguno	La definición de esta propiedad no tiene efecto en las aplicaciones que se ejecutan en el perfil de escritorio.
NativeProcess	`NativeProcess.isSupported`	La interacción con el proceso nativo sólo está disponible en las aplicaciones de perfil de escritorio ampliadas.
`Stage.orientation`	`Stage.supportsOrientationChange`	La API de orientación del escenario sólo está disponible en el perfil de dispositivo móvil.

Escritorio ampliado

Las aplicaciones de perfil de escritorio ampliadas tienen acceso a las mismas APIs de ActionScript que la aplicación de perfil de escritorio. Sin embargo, cuentan con dos capacidades adicionales:

- Las aplicaciones de perfil de escritorio ampliadas pueden iniciar procesos nativos e interaccionar con ellos. Para obtener más información, consulte una de las siguientes referencias:
  - Communicating with native processes in AIR (Comunicación con procesos nativos en AIR; en inglés.) (Para desarrolladores de HTML)
  - Communicating with native processes in AIR (Comunicación con procesos nativos en AIR; en inglés.) (Para desarrolladores de HTML)
- Las aplicación de perfil de escritorio ampliada puede llamar a `File.openWithDefaultApplication()` para los archivos de cualquier tipo. Las restricciones que se aplican en el perfil de escritorio no se aplican en el perfil de escritorio ampliado. El método `File.openWithDefaultApplication()` permite la apertura en la aplicación predeterminada registrada en el sistema operativo para abrir el archivo.

Dispositivo móvil

Las aplicaciones de perfil de dispositivo móvil tienen acceso la mayoría de las APIs de ActionScript 3.0, pero existen excepciones y limitaciones. Asimismo, existen APIs que sólo funcionan en aplicaciones de perfil de dispositivo móvil. Para obtener más información, consulte Compatibilidad de la API de ActionScript 3.0 con aplicaciones para iPhone.

Especificación de perfil al depurar con ADL

ADL comprueba si se especifican los perfiles admitidos en el elemento `supportedProfiles` del archivo descriptor de la aplicación. Si es así, ADL utilizará de forma predeterminada el primer perfil admitido incluido como perfil en el proceso de depuración.

Es posible especificar un perfil para la sesión de depuración de ADL utilizando el argumento de la línea de comandos `-profile`. (Consulte "Limitación de los perfiles de la aplicación de destino" en la página 77.) Este argumento se puede utilizar tanto si se especifica como si no un perfil en el elemento `supportedProfiles` en el archivo descriptor de la aplicación. No obstante, si se especifica un elemento `supportedProfiles`, es necesario incluir el perfil en la línea de comandos. De lo contrario, ADL genera un error.

Capítulo 13: Distribución, instalación y ejecución de aplicaciones de AIR

Las aplicaciones de AIR se distribuyen como un solo archivo de instalación de AIR que contiene el código de la aplicación y todos los componentes. Este archivo puede distribuirse por cualquiera de las vías tradicionales, por ejemplo descargándolo, por correo electrónico o en soportes físicos como CD-ROM. El usuario pueden instalar la aplicación haciendo doble clic en el archivo de AIR. Se puede utilizar la función de *instalación integrada*, que permite al usuario instalar la aplicación de AIR (y Adobe® AIR™, si es preciso) haciendo clic en un solo enlace en una página web.

Antes de poder distribuirlo, el archivo de instalación de AIR se debe empaquetar y firmar con un certificado de firma de código y una clave privada. La firma digital del archivo de instalación ofrece la seguridad de que la aplicación no ha sido modificada desde que se firmó. Además, si una entidad emisora de certificados fidedigna emite un certificado digital, los usuarios de su aplicación podrán confirmar su identidad como editor y firmante. El archivo de AIR se firma cuando se empaqueta la aplicación junto con la herramienta AIR Developer Tool (ADT).

Para obtener información sobre cómo empaquetar una aplicación en un archivo de AIR, consulte lo siguiente:

- Adobe® Flex® Builder™, consulte Empaquetado de aplicaciones de AIR con Flex Builder.
- Adobe® Flash® Builder™, consulte Empaquetado de aplicaciones de AIR con Flash Builder.
- Adobe® Flash® Professional, consulte Publicación para Adobe AIR.
- Adobe® Dreamweaver®, consulte Creación de una aplicación de AIR en Dreamweaver.
- SDK de Adobe® AIR®, consulte "Empaquetado de archivos de instalación de AIR con AIR Developer Tool (ADT)" en la página 54.

Instalación y ejecución de una aplicación de AIR desde el escritorio

Se puede simplemente enviar el archivo de AIR al destinatario. Por ejemplo: podría enviar el archivo de AIR como adjunto a un correo electrónico o como vínculo en una página web.

Una vez que descargó la aplicación de AIR, el usuario sigue estas instrucciones para instalarla:

1 Haga doble clic en el archivo de AIR.

Adobe AIR ya debe estar instalado en el ordenador.

2 En la ventana Instalación, deje seleccionada la configuración predeterminada y haga clic en Continuar.

En Windows, AIR realiza automáticamente lo siguiente:

- Instala la aplicación en el directorio Archivos de programa.
- Crea un acceso directo para la aplicación en el escritorio.
- Crea un acceso directo en el menú Inicio.
- Añade una entrada para la aplicación en Agregar o quitar programas, en el Panel de control.

En Mac OS, la aplicación se añade de forma predeterminada al directorio Aplicaciones.

Si la aplicación ya está instalada, el instalador ofrece al usuario la opción de abrir la versión existente de la misma o actualizarla a la versión del archivo de AIR descargado. El instalador identifica la aplicación utilizando para ello el ID de la aplicación y el ID del editor que figuran en el archivo de AIR.

3 Una vez concluida la instalación, haga clic en Finalizar.

En Mac OS, para instalar una versión actualizada de una aplicación el usuario debe contar con privilegios del sistema adecuados para instalar programas en el directorio de aplicaciones. En Windows y Linux, el usuario debe contar con privilegios de administrador.

Una aplicación también puede instalar una nueva versión mediante ActionScript o JavaScript. Para obtener más información, consulte "Actualización de aplicaciones de AIR" en la página 105.

Una vez instalada la aplicación de AIR, basta con que el usuario haga doble clic en la aplicación para ejecutarla, al igual que con cualquier otra aplicación del escritorio.

- En Windows, haga doble clic en el icono de la aplicación (que está instalada en el escritorio o en una carpeta) o seleccione la aplicación en el menú Inicio.
- En Linux, haga doble clic en el icono de la aplicación (que está instalada en el escritorio o en una carpeta) o seleccione la aplicación en el menú de aplicaciones.
- En Mac OS, haga doble clic en la aplicación en la carpeta en la que se instaló. El directorio de instalación predeterminado es /Aplicaciones.

La función de *instalación integrada* permite al usuario instalar una aplicación de AIR haciendo clic en un vínculo en una página web. La función de *invocación desde el navegador* permite al usuario ejecutar una aplicación de AIR instalada haciendo clic en un vínculo en una página web. Estas funciones se describen en la siguiente sección.

Instalación y ejecución de aplicaciones de AIR desde una página web

La función de instalación integrada permite incorporar un archivo SWF en una página web mediante la cual el usuario puede instalar una aplicación de AIR desde el navegador. Si no está instalado el motor de ejecución, lo instalará la función de instalación integrada. La función de instalación integrada permite al usuario instalar la aplicación de AIR sin necesidad de guardar el archivo de AIR en el ordenador.

Los SDK de AIR y Flex incluyen un archivo badge.swf que permite utilizar fácilmente la función de instalación integrada. Para obtener más información, consulte "Utilización del archivo badge.swf para instalar una aplicación de AIR" en la página 88.

Para ver una demostración del modo de uso de la función de instalación integrada, consulte el artículo de ejemplo de la guía de inicio rápido Distributing an AIR Application via the Web (http://www.adobe.com/go/learn_air_qs_seamless_install_es) (Distribución de una aplicación de AIR a través de la Web) (en inglés).

Personalización del archivo badge.swf de instalación integrada

Además de utilizar el archivo badge.swf suministrado con el SDK, puede crear su propio archivo SWF para usarlo en una página del navegador. El archivo SWF personalizado puede interactuar con el motor de ejecución de cualquiera de las formas siguientes:

- Puede instalar una aplicación de AIR. Consulte "Instalación de una aplicación de AIR desde el navegador" en la página 93.

- Puede comprobar si hay instalada una aplicación de AIR en particular. Consulte "Cómo comprobar desde una página web si una aplicación de AIR está instalada" en la página 92.
- Puede comprobar si está instalado el motor de ejecución. Consulte "Cómo comprobar si está instalado el motor de ejecución" en la página 92.
- Puede iniciar una aplicación de AIR instalada en el sistema del usuario. Consulte "Inicio desde el navegador de una aplicación de AIR instalada" en la página 94.

Estas capacidades se proporcionan al llamar a las API de un archivo SWF alojado en adobe.com: air.swf. Puede personalizar el archivo badge.swf y llamar a las API air.swf desde el propio archivo SWF.

Además, un archivo SWF que se ejecuta en el navegador puede comunicarse con una aplicación de AIR en curso utilizando la clase LocalConnection. Para obtener más información, consulte Comunicación con otras instancias de Flash Player y AIR (para desarrolladores de ActionScript) o Comunicación con otras instancias de Flash Player y AIR (para desarrolladores de HTML).

Importante: *las funciones que se describen en esta sección (y las API del archivo air.swf) requieren que el usuario tenga Adobe® Flash® Player 9 actualización 3 instalado en Windows o Mac OS. En Linux, la función de instalación integrada requiere Flash Player 10 (versión 10,0,12,36 o posterior). Se puede escribir código para comprobar la versión instalada de Flash Player y proveer una interfaz alternativa para el usuario si no está instalada la versión de Flash Player que se requiere. Por ejemplo, si hay una versión anterior de Flash Player instalada, puede proporcionar un vínculo a la versión de descarga del archivo de AIR (en lugar de utilizar el archivo badge.swf o la API air.swf para instalar una aplicación).*

Utilización del archivo badge.swf para instalar una aplicación de AIR

Los SDK de AIR y Flex incluyen un archivo badge.swf que permite utilizar fácilmente la función de instalación integrada. El archivo badge.swf puede instalar el motor de ejecución y una aplicación de AIR desde un vínculo en una página web. El archivo badge.swf y su código fuente se le proporcionan para que los distribuya a través de su sitio web.

Incorporación del archivo badge.swf en una página web

1. Localice los siguientes archivos, incluidos en el directorio samples/badge de los SDK de AIR y Flex, y añádalos a su servidor web.
   - badge.swf
   - default_badge.html
   - AC_RunActiveContent.js
2. Abra la página default_badge.html en un editor de textos.
3. En la página default_badge.html, en la función de JavaScript `AC_FL_RunContent()`, ajuste las definiciones del parámetro `FlashVars` para lo siguiente:

Parámetro	Descripción
`appname`	Nombre de la aplicación que muestra el archivo SWF si no está instalado el motor de ejecución.
`appurl`	(Obligatorio). URL del archivo de AIR que se va a descargar. Es necesario utilizar una URL absoluta (y no una relativa).
`airversion`	(Obligatorio). Para la versión 1.0 del motor de ejecución, defina esto en 1.0.
`imageurl`	URL de la imagen (opcional) a mostrar como logotipo.
`buttoncolor`	Color del botón de descarga (especificado como valor hexadecimal; por ejemplo, `FFCC00`).
`messagecolor`	Color del mensaje de texto que aparece debajo del botón si no está instalado el motor de ejecución (especificado como valor hexadecimal; por ejemplo, `FFCC00`).

4 El tamaño mínimo del archivo badge.swf es de 217 píxeles de anchura por 180 píxeles de altura. Ajuste los valores de los parámetros `width` y `height` de la función `AC_FL_RunContent()` de acuerdo con sus necesidades.

5 Cambie el nombre del archivo default_badge.html y ajuste su código (o inclúyalo en otra página HTML) para adaptarlo a sus necesidades.

Nota: *para la etiqueta `embed` HTML que carga el archivo badge.swf, no establezca el atributo `wmode`; déjelo definido en el valor predeterminado (`"window"`). Otras configuraciones de `wmode` evitarán la instalación en algunos sistemas. Asimismo, el uso de otras configuraciones de `wmode` genera un error: "Error #2044: ErrorEvent no controlado:. text=Error #2074: El escenario es demasiado pequeño para la interfaz de usuario de descarga."*

También se puede editar y recompilar el archivo badge.swf. Para obtener más información, consulte "Modificación del archivo badge.swf" en la página 89.

Instalación de la aplicación de AIR desde un vínculo de instalación integrada en una página web

Una vez que haya añadido a una página el vínculo de instalación integrada, el usuario podrá instalar la aplicación de AIR con sólo hacer clic en el vínculo del archivo SWF.

1 Examine la página HTML en un navegador web que disponga tenga Flash Player instalado (versión 9 actualización 3 o posterior en Windows y Mac OS, o bien, versión 10 en Linux).

2 En la página web, haga clic en el vínculo del archivo badge.swf.

- Si ha instalado el motor de ejecución, vaya al paso siguiente.
- Si no ha instalado el motor de ejecución, aparece un cuadro de diálogo que le pregunta si desea instalarlo. Instale el motor de ejecución (consulte "Instalación de Adobe AIR" en la página 6) y continúe con el siguiente paso.

3 En la ventana Instalación, deje seleccionada la configuración predeterminada y haga clic en Continuar.

En un equipo con Windows, AIR realiza automáticamente lo siguiente:

- Instala la aplicación en c:\Archivos de programa\.
- Crea un acceso directo para la aplicación en el escritorio.
- Crea un acceso directo en el menú Inicio.
- Añade una entrada para la aplicación en Agregar o quitar programas, en el Panel de control.

En Mac OS el instalador añade la aplicación al directorio de aplicaciones (por ejemplo, en el directorio /Aplicaciones de Mac OS).

En un ordenador con Linux, AIR realiza automáticamente lo siguiente:

- Instala la aplicación en /opt.
- Crea un acceso directo para la aplicación en el escritorio.
- Crea un acceso directo en el menú Inicio.
- Añade una entrada para la aplicación en el administrador del paquete del sistema.

4 Seleccione las opciones que desee y haga clic en el botón Instalar.

5 Una vez concluida la instalación, haga clic en Finalizar.

Modificación del archivo badge.swf

El SDK de Flex y AIR proporciona los archivos de origen para el archivo badge.swf. Estos archivos están incluidos en la carpeta samples/badge del SDK:

Archivos de origen	Descripción
badge.fla	El archivo de origen de Flash se utiliza para compilar el archivo badge.swf. El archivo badge.fla se compila en un archivo SWF 9 (que se puede cargar en Flash Player).
AIRBadge.as	Una clase de ActionScript 3.0 que define la clase de base que se utiliza en el archivo badge.fla.

Puede utilizar Flash CS3 o Flash CS4 para rediseñar la interfaz visual del archivo badge.swf.

La función constructora `AIRBadge()`, definida en la clase AIRBadge, carga el archivo air.swf alojado en http://airdownload.adobe.com/air/browserapi/air.swf. El archivo air.swf incluye código para utilizar la función de instalación integrada.

El método `onInit()` (en la clase AIRBadge) se invoca una vez satisfactoriamente cargado el archivo air.swf:

```
private function onInit(e:Event):void {
    _air = e.target.content;
    switch (_air.getStatus()) {
        case "installed" :
            root.statusMessage.text = "";
            break;
        case "available" :
            if (_appName && _appName.length > 0) {
                root.statusMessage.htmlText = "<p align='center'><font color='#"
                        + _messageColor + "'>In order to run " + _appName +
                        ", this installer will also set up Adobe® AIR®.</font></p>";
            } else {
                root.statusMessage.htmlText = "<p align='center'><font color='#"
                        + _messageColor + "'>In order to run this application, "
                        + "this installer will also set up Adobe® AIR®.</font></p>";
            }
            break;
        case "unavailable" :
            root.statusMessage.htmlText = "<p align='center'><font color='#"
                        + _messageColor
                        + "'>Adobe® AIR® is not available for your system.</font></p>";
            root.buttonBg_mc.enabled = false;
            break;
    }
}
```

El código define la variable global `_air` en la clase principal del archivo air.swf. Esta clase incluye los siguientes métodos públicos, a los que tiene acceso el archivo badge.swf para llamar a la función de instalación integrada:

Método	Descripción
`getStatus()`	Determina si el motor de ejecución está instalado (o puede instalarse) en el ordenador. Para obtener más información, consulte "Cómo comprobar si está instalado el motor de ejecución" en la página 92.
`installApplication()`	Instala la aplicación especificada en el equipo del usuario. Para obtener más información, consulte "Instalación de una aplicación de AIR desde el navegador" en la página 93. • `url`: una cadena que define la URL. Hay que utilizar una ruta de URL absoluta (y no una relativa). • `runtimeVersion`: una cadena que indica la versión del motor de ejecución (por ejemplo: `"1.0.M6"`) que requiere la aplicación a instalarse. • `arguments`: argumentos a pasarse a la aplicación si se inicia en cuanto se haya instalado. La aplicación se inicia en el momento de instalarla si el elemento `allowBrowserInvocation` está definido en `true` en el archivo descriptor de la aplicación. (Para obtener más información sobre el archivo descriptor de la aplicación, consulte "Configuración de las propiedades de una aplicación de AIR" en la página 71.) Si la aplicación se inicia a resultas de una instalación integrada desde el navegador (habiendo el usuario seleccionado iniciarla en cuanto se instalara), el objeto NativeApplication de la aplicación distribuye un objeto BrowserInvokeEvent sólo si se pasan argumentos. Tenga en cuenta las consecuencias para la seguridad que pueden tener los datos que pase a la aplicación. Para obtener más información, consulte "Inicio desde el navegador de una aplicación de AIR instalada" en la página 94.

Los valores para `url` y `runtimeVersion` se pasan al archivo SWF a través de las opciones de FlashVars en la página HTML contenedora.

Si la aplicación se inicia automáticamente al instalarla, se puede utilizar la comunicación por LocalConnection para que la aplicación instalada se ponga en contacto con el archivo badge.swf al invocarse. Para obtener más informacióm, consulte Comunicación con otras instancias de Flash Player y AIR (para desarrolladores de ActionScript) o Comunicación con otras instancias de Flash Player y AIR (para desarrolladores de HTML).

También se puede llamar al método `getApplicationVersion()` del archivo air.swf para comprobar si una aplicación está instalada. Se puede llamar a este método antes del proceso de instalación de la aplicación o una vez iniciada la instalación. Para obtener más información, consulte "Cómo comprobar desde una página web si una aplicación de AIR está instalada" en la página 92.

Carga del archivo air.swf

Puede crear su propio archivo SWF que utilice las API del archivo air.swf para interactuar con el motor de ejecución y las aplicaciones de AIR desde una página web en un navegador. El archivo air.swf está alojado en http://airdownload.adobe.com/air/browserapi/air.swf. Para hacer referencia a las API de air.swf desde el archivo SWF, cargue el archivo air.swf en el mismo dominio de la aplicación que el archivo SWF. El código siguiente muestra un ejemplo de cargar el archivo air.swf en el dominio de la aplicación del archivo SWF de carga:

```
var airSWF:Object; // This is the reference to the main class of air.swf
var airSWFLoader:Loader = new Loader(); // Used to load the SWF
var loaderContext:LoaderContext = new LoaderContext();
                                // Used to set the application domain

loaderContext.applicationDomain = ApplicationDomain.currentDomain;

airSWFLoader.contentLoaderInfo.addEventListener(Event.INIT, onInit);
airSWFLoader.load(new URLRequest("http://airdownload.adobe.com/air/browserapi/air.swf"),
                  loaderContext);

function onInit(e:Event):void
{
    airSWF = e.target.content;
}
```

Una vez cargado el archivo air.swf (cuando el objeto `contentLoaderInfo` de Loader distribuye el evento `init`), se puede llamar a cualquiera de las API de air.swf. descritas en las siguientes secciones.

Nota: *el archivo badge.swf suministrado con los SDK de AIR y Flex carga automáticamente el archivo air.swf. Consulte "Utilización del archivo badge.swf para instalar una aplicación de AIR" en la página 88. Las instrucciones que aparecen en esta sección son para crear su propio archivo SWF que cargue el archivo air.swf.*

Cómo comprobar si está instalado el motor de ejecución

Un archivo SWF puede comprobar si el motor de ejecución está instalado, llamando al método `getStatus()` en el archivo air.swf cargado desde http://airdownload.adobe.com/air/browserapi/air.swf. Para obtener más información, consulte "Carga del archivo air.swf" en la página 91.

Una vez cargado el archivo air.swf, el archivo SWF puede llamar al método `getStatus()` del archivo air.swf como en el ejemplo siguiente:

```
var status:String = airSWF.getStatus();
```

El método `getStatus()` devuelve uno de los siguientes valores de cadena, basado en el estado del motor de ejecución en el ordenador:

Valor de la cadena	Descripción
`"available"`	El motor de ejecución puede instalarse en este ordenador pero ahora no está instalado.
`"unavailable"`	El motor de ejecución no puede instalarse en este ordenador.
`"installed"`	El motor de ejecución está instalado en este ordenador.

El método `getStatus()` emite un error si la versión necesaria de Flash Player (versión 9 actualización 3 o posterior en Windows y Mac OS, o bien, versión 10 en Linux) no está instalada en el navegador.

Cómo comprobar desde una página web si una aplicación de AIR está instalada

Un archivo SWF puede comprobar si una aplicación de AIR (con ID de la aplicación e ID del editor que coincidan) está instalada, llamando al método `getApplicationVersion()` en el archivo air.swf cargado desde http://airdownload.adobe.com/air/browserapi/air.swf. Para obtener más información, consulte "Carga del archivo air.swf" en la página 91.

Una vez cargado el archivo air.swf, el archivo SWF puede llamar al método `getApplicationVersion()` del archivo air.swf como en el ejemplo siguiente:

```
 var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
airSWF.getApplicationVersion(appID, pubID, versionDetectCallback);

function versionDetectCallback(version:String):void
{
    if (version == null)
    {
        trace("Not installed.");
        // Take appropriate actions. For instance, present the user with
        // an option to install the application.
    }
    else
    {
        trace("Version", version, "installed.");
        // Take appropriate actions. For instance, enable the
        // user interface to launch the application.
    }
}
```

El método `getApplicationVersion()` cuenta con los siguientes parámetros:

Parámetros	Descripción
`appID`	ID de la aplicación. Para obtener más información, consulte "Definición de la identidad de la aplicación" en la página 74.
`pubID`	ID del editor de la aplicación. Para obtener más información, consulte "Identificador del editor de AIR" en la página 98. Si la aplicación en cuestión no dispone de un ID de editor, establezca el parámetro `pubID` en una cadena vacía ("").
`callback`	Función callback que cumple la función de controlador. El método getApplicationVersion() funciona de modo asíncrono, y al detectar la versión instalada (o la falta de una), se invoca este método callback. La definición del método callback debe incluir un parámetro, una cadena de caracteres, que se establece como la cadena de la versión de la aplicación instalada. Si la aplicación no está instalada, se pasa un valor nulo a la función, como se muestra en el ejemplo de código anterior.

El método `getApplicationVersion()` emite un error si la versión necesaria de Flash Player (versión 9 actualización 3 o posterior en Windows y Mac OS, o bien, versión 10 en Linux) no está instalada en el navegador.

Nota: *a partir de AIR 1.5.3, el ID de editor queda desfasado. Los ID de editor no se vuelven a asignar a una aplicación de forma automática. Por compatibilidad con versiones anteriores, las aplicaciones pueden continuar para especificar un ID de editor.*

Instalación de una aplicación de AIR desde el navegador

Un archivo SWF puede instalar una aplicación de AIR, llamando al método `installApplication()` en el archivo air.swf cargado desde `http://airdownload.adobe.com/air/browserapi/air.swf`. Para obtener más información, consulte "Carga del archivo air.swf" en la página 91.

Una vez cargado el archivo air.swf, el archivo SWF puede llamar al método `installApplication()` del archivo air.swf como en el código siguiente:

```
var url:String = "http://www.example.com/myApplication.air";
var runtimeVersion:String = "1.0";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.installApplication(url, runtimeVersion, arguments);
```

El método `installApplication()` instala la aplicación especificada en el equipo del usuario. Este método utiliza los siguientes parámetros:

Parámetro	Descripción
`url`	Una cadena de caracteres que define la URL del archivo de AIR a instalar. Hay que utilizar una ruta de URL absoluta (y no una relativa).
`runtimeVersion`	Una cadena que indica la versión del motor de ejecución (por ejemplo: "1.0") que requiere la aplicación a instalarse.
`arguments`	Un conjunto de argumentos a pasarse a la aplicación si se inicia en cuanto se haya instalado. Únicamente los caracteres alfanuméricos se reconocen en los argumentos. Si necesita transmitir otros valores, considere el uso de un esquema de codificación. La aplicación se inicia en el momento de instalarla si el elemento `allowBrowserInvocation` está definido en `true` en el archivo descriptor de la aplicación. (Para obtener más información sobre el archivo descriptor de la aplicación, consulte "Configuración de las propiedades de una aplicación de AIR" en la página 71.) Si la aplicación se inicia a resultas de una instalación integrada desde el navegador (habiendo el usuario seleccionado iniciarla en cuanto se instalara), el objeto NativeApplication de la aplicación distribuye un objeto BrowserInvokeEvent sólo si se pasan argumentos. Para obtener más información, consulte "Inicio desde el navegador de una aplicación de AIR instalada" en la página 94.

El método `installApplication()` sólo funciona cuando se le llama en el controlador de eventos para un evento de usuario, por ejemplo al hacer clic con el ratón.

El método `installApplication()` emite un error si la versión necesaria de Flash Player (versión 9 actualización 3 o posterior en Windows y Mac OS, o bien, versión 10 en Linux) no está instalada en el navegador.

En Mac OS, para instalar una versión actualizada de una aplicación el usuario debe contar con privilegios del sistema adecuados para instalar programas en el directorio de aplicaciones (y privilegios de administrador si la aplicación actualiza el motor de ejecución). En Windows, el usuario debe contar con privilegios de administrador.

También se puede llamar al método `getApplicationVersion()` del archivo air.swf para comprobar si una aplicación ya está instalada. Se puede llamar a este método antes de que empiece el proceso de instalación de la aplicación o una vez iniciada la instalación. Para obtener más información, consulte "Cómo comprobar desde una página web si una aplicación de AIR está instalada" en la página 92. Una vez en ejecución la aplicación, ésta puede comunicarse con el contenido SWF en el navegador utilizando la clase LocalConnection. Para obtener más informaciom, consulte Comunicación con otras instancias de Flash Player y AIR (para desarrolladores de ActionScript) o Comunicación con otras instancias de Flash Player y AIR (para desarrolladores de HTML).

Inicio desde el navegador de una aplicación de AIR instalada

Para utilizar la función de invocación desde el navegador (lo que permite iniciar la aplicación desde el navegador), el archivo descriptor de la aplicación en cuestión debe incluir la siguiente definición:

```
<allowBrowserInvocation>true</allowBrowserInvocation>
```

Para obtener más información sobre el archivo descriptor de la aplicación, consulte "Configuración de las propiedades de una aplicación de AIR" en la página 71.

Un archivo SWF en el navegador puede iniciar una aplicación de AIR, llamando al método `launchApplication()` en el archivo air.swf cargado desde http://airdownload.adobe.com/air/browserapi/air.swf. Para obtener más información, consulte "Carga del archivo air.swf" en la página 91.

Una vez cargado el archivo air.swf, el archivo SWF puede llamar al método `launchApplication()` del archivo air.swf como en el código siguiente:

```
var appID:String = "com.example.air.myTestApplication";
var pubID:String = "02D88EEED35F84C264A183921344EEA353A629FD.1";
var arguments:Array = ["launchFromBrowser"]; // Optional
airSWF.launchApplication(appID, pubID, arguments);
```

El método `launchApplication()` se define en el nivel superior del archivo air.swf (que se carga en el dominio de aplicación de la interfaz de usuario del archivo SWF). Al llamar a este método, AIR inicia la aplicación especificada (si está instalada y se permite la invocación desde el navegador mediante la opción `allowBrowserInvocation` del archivo descriptor de la aplicación). El método utiliza los siguientes parámetros:

Parámetro	Descripción
`appID`	ID de la aplicación que se va a iniciar. Para obtener más información, consulte "Definición de la identidad de la aplicación" en la página 74.
`pubID`	ID del editor de la aplicación que se va a iniciar. Para obtener más información, consulte "Identificador del editor de AIR" en la página 98. Si la aplicación en cuestión no dispone de un ID de editor, establezca el parámetro `pubID` en una cadena vacía ("").
`arguments`	Conjunto de argumentos que se transmiten a la aplicación. El objeto NativeApplication de la aplicación distribuye un evento BrowserInvokeEvent que tiene una propiedad "arguments" definida en este conjunto.. Únicamente los caracteres alfanuméricos se reconocen en los argumentos. Si necesita transmitir otros valores, considere el uso de un esquema de codificación.

El método `launchApplication()` sólo funciona cuando se le llama en el controlador de eventos para un evento de usuario, por ejemplo al hacer clic con el ratón.

El método `launchApplication()` emite un error si la versión necesaria de Flash Player (versión 9 actualización 3 o posterior en Windows y Mac OS, o bien, versión 10 en Linux) no está instalada en el navegador.

Si el elemento `allowBrowserInvocation` está definido en `false` en el archivo descriptor de la aplicación, la llamada al método `launchApplication()` no surtirá efecto.

Antes de presentar la interfaz de usuario para iniciar la aplicación, puede ser conveniente llamar al método `getApplicationVersion()` en el archivo air.swf. Para obtener más información, consulte "Cómo comprobar desde una página web si una aplicación de AIR está instalada" en la página 92.

Cuando se invoca la aplicación a través de la función de invocación desde el navegador, el objeto NativeApplication de la aplicación distribuye un objeto BrowserInvokeEvent. Para obtener información, consulte Invocación de una aplicación de AIR desde el navegador (para desarrolladores de ActionScript), o bien, Invocación de una aplicación de AIR desde el navegador (para desarrolladores de HTML).

Si utiliza la función de invocación desde el navegador, asegúrese de tener en cuenta las posibles consecuencias para la seguridad. Estas implicaciones se describen en Invocación de una aplicación de AIR desde el navegador (para desarrolladores de ActionScript) e Invocación de una aplicación de AIR desde el navegador (para desarrolladores de HTML).

Una vez en ejecución la aplicación, ésta puede comunicarse con el contenido SWF en el navegador utilizando la clase LocalConnection. Para obtener más informaciom, consulte Comunicación con otras instancias de Flash Player y AIR (para desarrolladores de ActionScript) o Comunicación con otras instancias de Flash Player y AIR (para desarrolladores de HTML).

Nota: *a partir de AIR 1.5.3, el ID de editor queda desfasado. Los ID de editor no se vuelven a asignar a una aplicación de forma automática. Por compatibilidad con versiones anteriores, las aplicaciones pueden continuar para especificar un ID de editor.*

Implementación en la empresa

Los administradores de TI pueden instalar el motor de ejecución de Adobe AIR y aplicaciones de AIR de forma silenciosa con herramientas de implementación estándar. Los administradores de TI pueden llevar a cabo las siguientes tareas:

- Realizar una instalación silenciosa del motor de ejecución de Adobe AIR empleando herramientas como Microsoft SMS, IBM Tivoli o cualquier herramienta de implementación que permita las instalaciones silenciosas que utilizan un arrancador.
- Efectuar una instalación silenciosa de la aplicación de AIR con las mismas herramientas que se utilizan para implementar el motor de ejecución.

Para obtener más información, consulte *Adobe AIR Administrator's Guide (http://www.adobe.com/go/learn_air_admin_guide_en).*

Registros de instalación

Los registros de instalación se crean cuando se instala el propio motor de ejecución de AIR o una aplicación de AIR. Los archivos de registro se pueden examinar para ayudar a determinar la causa de cualquier problema de actualización o instalación que se produzca.

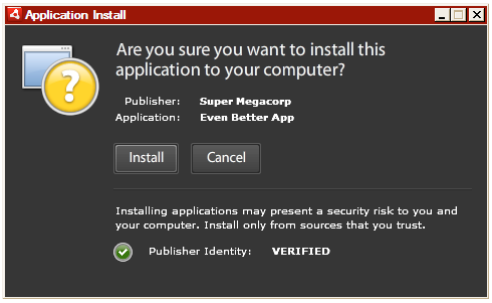
Los archivos de registro se crean en las siguientes ubicaciones:

- Mac: registro del sistema estándar(`/private/var/log/system.log`)
- Windows XP: `C:\Documents and Settings\<username>\Configuración local\Datos de programa\Adobe\AIR\logs\Install.log`
- Windows Vista, Windows 7: `C:\Users\<username>\AppData\Local\Adobe\AIR\logs\Install.log`
- Linux: `/home/<username>/.appdata/Adobe/AIR/Logs/Install.log`

Nota: *estos archivos de registro no se crearon en versiones de AIR anteriores a AIR 2.*

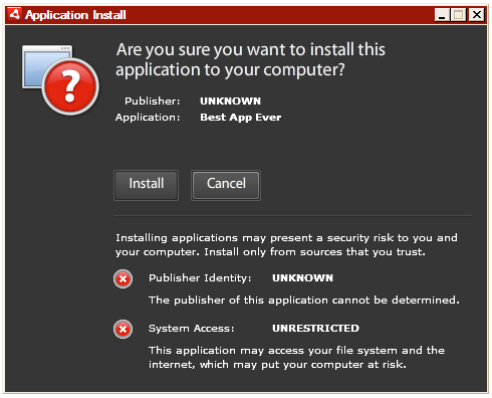
Firma digital de archivos de AIR

La firma digital de sus archivos de instalación de AIR con un certificado proporcionado por una entidad emisora de certificados (AC) reconocida ofrece seguridad a los usuarios de que la aplicación que están instalando no ha sido alterada ni de modo accidental ni malintencionado e identifica a usted como el firmante (editor). Si la aplicación de AIR está firmada con un certificado de confianza, o está *encadenada* con un certificado de confianza en el ordenador de instalación, AIR presenta el nombre del editor durante la instalación.



Cuadro de diálogo de confirmación de instalación firmado por un certificado de confianza

Si una aplicación se firma con un certificado con firma automática (o un certificado que no se vincula a un certificado de confianza), el usuario debe asumir un mayor riesgo de seguridad al instalar la aplicación. El cuadro de diálogo de instalación refleja este riesgo adicional:



Cuadro de diálogo de confirmación de instalación firmado por un certificado con firma automática

***Importante:** una entidad malintencionada podría falsificar un archivo de AIR con su identidad si lograra obtener su archivo de almacén de claves para la firma descubre si clave privada.*

Certificados con firma de código

Las garantías de seguridad, limitaciones y obligaciones legales respecto del uso de certificados de firma de código se reseñan en la declaración de prácticas de certificación (DPC) y los acuerdos de suscriptor publicados por la autoridad de certificación emisora. Para obtener más información sobre los acuerdos para las entidades emisoras de certificados que actualmente proporcionan certificados de firma de código de AIR, consulte:

ChosenSecurity (http://www.chosensecurity.com/products/tc_publisher_id_adobe_air.htm) (en inglés)

ChosenSecurity CPS (http://www.chosensecurity.com/resource_center/repository.htm) (en inglés)

GlobalSign (http://www.globalsign.com/developer/code-signing-certificate/index.htm) (en inglés)

GlobalSign CPS (http://www.globalsign.com/repository/index.htm) (en inglés)

Thawte CPS (http://www.thawte.com/cps/index.html) (en inglés)

Thawte Code Signing Developer's Agreement (Acuerdo de desarrollador para firma de códigos de Thawte) (http://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/develcertsign.pdf) (en inglés)

VeriSign CPS (http://www.verisign.com/repository/CPS/) (en inglés)

Acuerdo de suscripción de Verisign (https://www.verisign.es/repository/subscriber/SUBAGR.html)

Firma de códigos de AIR

Cuando se firma un archivo de AIR se incluye una firma digital en el archivo de instalación. La firma incluye un resumen del paquete, el cual se utiliza para verificar que el archivo de AIR no se haya alterado desde que se firmó, e incluye información acerca del certificado de firma, que sirve para verificar la identidad del editor.

AIR utiliza la infraestructura de claves públicas (PKI), compatible a través del almacén de certificados del sistema operativo, para establecer si un certificado es de confianza. El ordenador en el que se instala una aplicación de AIR debe o bien confiar directamente en el certificado que se utiliza para firmar la aplicación de AIR, o bien confiar en una serie de certificados que vinculan el certificado con una entidad emisora de certificados de confianza para que se pueda verificar la información del editor.

Si se firma un archivo de AIR con un certificado que no está encadenado con uno de los certificados raíz de confianza (que normalmente incluye a todos los certificados con firma automática), no se puede verificar la información del editor. Si bien AIR puede determinar que un paquete de AIR no ha sido alterado desde que se firmó, no hay manera de saber quién creó y firmó el archivo.

Nota: *un usuario puede optar por confiar en un certificado con firma automática y en adelante las aplicaciones de AIR firmadas con el certificado presentarán el valor del campo de nombre común en el certificado como nombre del editor. AIR no proporciona ningún mecanismo para que un usuario designe un certificado como "de confianza". El certificado (sin incluir la clave privada) debe proporcionarse por separado al usuario, quien deberá emplear uno de los mecanismos suministrados con el sistema operativo o una herramienta adecuada para importar el certificado en el lugar correcto en el almacén de certificados del sistema.*

Identificador del editor de AIR

Importante: *a partir de AIR 1.5.3, el ID de editor queda desfasado y no se vuelve a calcular en función del certificado de firma de código. Las nuevas aplicaciones no necesitan ni deben utilizar un ID de editor. Al actualizar las aplicaciones existentes, debe especificar el ID de editor original en el archivo descriptor de la aplicación.*

Antes de AIR 1.5.3, el instalador de aplicaciones de AIR generó un ID de editor durante la instalación de un archivo de AIR. Se trataba de un identificador exclusivo del certificado que se utiliza para firmar el archivo de AIR. Si se vuelve a utilizar el mismo certificado para varias aplicaciones de AIR, todas tendrán el mismo ID de editor. La firma de una actualización de la aplicación con un certificado diferente y en ocasiones incluso con una instancia renovada del certificado original cambió el ID de editor.

En AIR 1.5.3 y versiones posteriores, un ID de editor no se asigna mediante AIR. Una aplicación publicada con AIR 1.5.3 puede especificar una cadena de ID de editor en el descriptor de la aplicación. Únicamente es necesario especificar un ID de editor cuando se publiquen actualizaciones para aplicaciones publicadas en un principio para versiones de AIR antes de 1.5.3. Si no se especifica el ID original en el descriptor de la aplicación, el nuevo paquete de AIR no se tratará como una actualización de la aplicación existente.

Para determinar el ID de editor original, localice el archivo `publisherid` en el subdirectorio META-INF/AIR donde se instaló la aplicación original. La cadena de este archivo es el ID de editor. El descriptor de la aplicación debe especificar el motor de ejecución de AIR 1.5.3 (o posterior) en la declaración del espacio de nombres del archivo descriptor de la aplicación con el fin de especificar el ID de editor manualmente.

El ID de editor, cuando se encuentra presente, se emplea para lo siguiente:

- Como parte de la clave de cifrado para el almacén local cifrado.
- Como parte de la ruta para el directorio de almacenamiento de la aplicación.

- Como parte de la cadena de conexión para conexiones locales.
- Como parte de la cadena de identidad utilizada para invocar una aplicación la API en navegador de AIR.
- Como parte de OSID (utilizado al crear programas personalizados de instalación y desinstalación).

Cuando cambia un ID de editor, el comportamiento de cualquier función de AIR basada en el ID también cambia. Por ejemplo, ya no se puede acceder a los datos del almacén local cifrado existente y cualquier instancia de Flash o AIR que cree una conexión local con la aplicación debe utilizar el nuevo ID en la cadena de conexión. El ID de editor para una aplicación instalada no puede cambiar en AIR 1.5.3 ni en versiones posteriores. Si se utiliza un ID de editor diferente al publicar un paquete de AIR, el instalador trata el nuevo paquete como una aplicación distinta en lugar de como una actualización.

Formatos de certificados

Las herramientas de firma de AIR aceptan cualquier almacén de claves accesibles a través de la arquitectura de criptografía de Java (Java Cryptography Architecture o JCA). En esto se incluyen los almacenes de claves basados en archivos, como archivos de formato PKCS12 (que suelen tener la extensión de archivo .pfx o .p12), archivos .keystore de Java, almacenes de claves de hardware PKCS11 y los almacenes de claves del sistema. Los formatos de almacenes de claves a los que tiene acceso ADT depende de la versión y configuración del motor de ejecución de Java que se utilice para ejecutar ADT. El acceso a algunos tipos de almacén de claves, como los token de hardware PKCS11, pueden necesitar que se instalen y configuren plugins de JCA y controladores de software adicionales.

Para firmar archivos de AIR, puede utilizar la mayoría de los certificados de firma de código existentes, o bien, puede obtener un certificado nuevo emitido expresamente para firmar aplicaciones de AIR. Se pueden utilizar, por ejemplo, cualquiera de los siguientes tipos de certificados de Verisign, Thawte, GlobalSign o ChosenSecurity:

- ChosenSecurity (en inglés)
  - TC Publisher ID para Adobe AIR
- GlobalSign (en inglés)
  - Certificado de firma de código ObjectSign
- Thawte:
  - Certificado de AIR Developer
  - Certificado de Apple Developer
  - Certificado de JavaSoft Developer
  - Certificado de Microsoft Authenticode
- VeriSign (en inglés):
  - ID digital de Adobe AIR
  - Microsoft Authenticode Digital ID
  - Sun Java Signing Digital ID

Nota: *el certificado se debe crear para realizar la firma de código. No se puede utilizar un certificado SSL ni otro tipo de certificado para firmar archivos de AIR.*

Marcas de hora

Cuando se firma un archivo de AIR, la herramienta de empaquetado consulta al servidor de una autoridad de marcas de hora para obtener una fecha y hora de la firma que sean verificables independientemente. La marca de hora que se obtiene se incorpora en el archivo de AIR. Siempre y cuando el certificado de firma sea válido en el momento de firmarlo, se podrá instalar el archivo de AIR, incluso después de caducado el certificado. Por otro lado, si no se obtiene la marca de hora, el archivo de AIR dejará de poder instalarse cuando caduque o se revoque el certificado.

La opción predeterminada es que las herramientas de empaquetado de AIR obtienen una marca de hora. No obstante, para que las aplicaciones puedan empaquetarse cuando no se dispone del servicio de marcas de hora, el marcado de hora se puede desactivar. Adobe recomienda que todo archivo de AIR que se distribuya al público incluya una marca de hora.

La autoridad de marcas de hora predeterminada que utilizan las herramientas de empaquetado de AIR es Geotrust.

Obtención de un certificado

Para obtener un certificado, normalmente visitaría el sitio web de la entidad emisora de certificados y llevaría a cabo el proceso de adquisición de la empresa en cuestión. Las herramientas que se utilizan para producir el archivo del almacén de claves que necesitan las herramientas de AIR dependen del tipo de certificado que se compre, cómo se guarda el certificado en el ordenador receptor y, en algunos casos, el navegador que se utilizó para obtener el certificado. Por ejemplo, para obtener y exportar un certificado de Adobe Developer desde Thawte, debe utilizar Mozilla Firefox. El certificado podrá entonces exportarse como archivo con prefijo .pfx o .p12 directamente desde la interfaz de usuario de Firefox.

Nota: *las versiones de Java 1.5 y posteriores no aceptan caracteres ASCII superior en contraseñas utilizadas para proteger archivos de certificado PKCS12. Las herramientas de desarrollo de AIR utilizan Java para crear paquetes firmados de AIR. Si el certificado se exporta como archivo .p12 o .pfx, utilice únicamente caracteres ASCII normales en la contraseña.*

Se puede generar un certificado con firma automática con Air Development Tool (ADT) que se utiliza para empaquetar los archivos de instalación de AIR. También pueden utilizarse algunas herramientas de terceros.

Para ver las instrucciones sobre cómo generar un certificado con firma automática, así como instrucciones para firmar un archivo de AIR, consulte "Empaquetado de archivos de instalación de AIR con AIR Developer Tool (ADT)" en la página 54. También se pueden exportar y firmar archivos de AIR con Flash Builder, Dreamweaver y la actualización de AIR para Flash.

El siguiente ejemplo describe cómo obtener un certificado de desarrollador de AIR de la autoridad de certificación Thawte y prepararlo para el uso con ADT.

Ejemplo: Cómo obtener un certificado de desarrollador de AIR con Thawte

Nota: *este ejemplo muestra una sola de las diversas formas de obtener y preparar un certificado de firma de código para su uso. Cada entidad emisora de certificados cuenta con sus propias directivas y procedimientos.*

Para adquirir un certificado de desarrollador de AIR, el sitio web de Thawte exige el uso del navegador Mozilla Firefox. La clave privada para el certificado se guarda en el almacén de claves del navegador. Asegúrese de que el almacén de claves de Firefox esté protegido con una contraseña maestra y que el ordenador mismo sea seguro desde el punto de vista físico. (Podrá exportar y eliminar el certificado y la clave privada del almacén de claves del navegador una vez finalizado el proceso de adquisición).

Como parte del proceso de inscripción para el certificado se genera un par de claves: pública y privada. La clave privada se guarda automáticamente en el almacén de claves de Firefox. Deberá utilizar el mismo ordenador y navegador para solicitar y recuperar el certificado del sitio web de Thawte.

1 Visite el sitio web de Thawte y vaya a la página de productos para Certificados para firma de códigos.

2 En la lista de certificados para firma de códigos, seleccione el certificado de Adobe AIR Developer.

3 Complete el proceso de inscripción de tres pasos. Necesitará facilitar información sobre su organización, así como datos de contacto. A continuación, Thawte lleva a cabo su proceso de verificación de identidad y es posible que solicite información suplementaria. Una vez finalizada la verificación, Thawte le enviará un correo electrónico con instrucciones sobre cómo recuperar el certificado.

Nota: *para obtener más información (en inglés) sobre el tipo de documentación que se requiere, haga clic en: https://www.thawte.com/ssl-digital-certificates/free-guides-whitepapers/pdf/enroll_codesign_eng.pdf.*

4 Recupere el certificado emitido del sitio de Thawte. El certificado se guarda automáticamente en el almacén de claves de Firefox.

5 Exporte del almacén de claves un archivo que contenga la clave privada y el certificado siguiendo los pasos que se indican a continuación:

Nota: *la clave privada y el certificado de Firefox se exportan en un formato .p12 (pfx) compatible con ADT, Flex, Flash y Dreamweaver.*

a Abra el cuadro de diálogo *Administrador de certificados* de Firefox:

b En Windows: abra Herramientas -> Opciones -> Opciones avanzadas -> Cifrado -> Ver certificados.

c En Mac OS: abra Firefox -> Preferencias -> Avanzado -> Cifrado -> Ver certificados.

d En Linux: abra Editar -> Preferencias -> Avanzado -> Cifrado -> Ver certificados.

e Seleccione el certificado para firma de códigos de Adobe AIR en la lista de certificados y haga clic en el botón **Hacer copia**.

f Escriba un nombre de archivo y el lugar al que se debe exportar el archivo del almacén de claves, y haga clic en **Guardar**.

g Si utilizó la contraseña maestra de Firefox, se le solicitará escribir la contraseña para el dispositivo de seguridad del software a fin de poder exportar el archivo. (Esta contraseña es de uso exclusivo en Firefox).

h En el cuadro de diálogo para la *contraseña para la copia de seguridad del certificado*, cree una contraseña para el archivo del almacén de claves.

Importante: *esta contraseña protege el archivo del almacén de claves y se necesita cuando se utiliza el archivo para firmar aplicaciones de AIR. Conviene elegir una contraseña segura.*

i Haga clic en OK (Aceptar). Debería recibir un mensaje confirmando la creación de la contraseña para copias de seguridad. El archivo del almacén contiene la clave privada y el certificado se guarda con una extensión .p12 (en formato PKCS12).

6 Utilice el archivo de almacén de claves exportado con ADT, Flash Builder, Flash Professional o Dreamweaver. Hay que utilizar la contraseña creada para el archivo cada vez que se firme una aplicación de AIR.

Importante: *la clave privada y el certificado se siguen guardando en el almacén de claves de Firefox. Mientras que esto permite exportar otra copia del archivo del certificado, también facilita otro punto de acceso que debe protegerse para mantener la seguridad de su certificado y su clave privada.*

Cambio de certificado

En algunos casos, es necesario cambiar el certificado utilizado para firmar actualizaciones de la aplicación de AIR. Entre estos casos se encuentran:

- Renovación del certificado de firma original.

- La actualización de un certificado con firma automática a un certificado emitido por una entidad emisora de certificados.
- El cambio de un certificado con firma automática que va a caducar por otro.
- El cambio de un certificado comercial a otro (por ejemplo, cuando cambia la identidad de la empresa).

Para que AIR reconozca un archivo de AIR como actualización, es necesario firmar tanto los archivos de AIR originales como de actualización con el mismo certificado, o bien, aplicar una firma de migración de certificados a la actualización. Una firma de migración es una segunda firma aplicada al paquete de AIR de actualización utilizando el certificado original. La firma de migración utiliza el certificado original, que establece que el firmante es el editor original de la aplicación.

Una vez instalado un archivo de AIR con una firma de migración, el nuevo certificado pasa a ser el certificado principal. Las actualizaciones posteriores no requieren una firma de migración. No obstante, las firmas de migración se deben aplicar tanto tiempo como sea posible a los usuarios que suelan omitir las actualizaciones.

Importante: *el certificado se deben cambiar antes de que caduque el certificado original. Si no se crea una actualización firmada con firma de migración antes de que caduque el certificado, los usuarios tendrán que desinstalar la versión existente de la aplicación antes de instalar la actualización. A partir de AIR 1.5.3, un certificado caducado se puede utilizar para aplicar una firma de migración en un período de gracia de 180 días una vez haya caducado el certificado. (El certificado caducado no se puede utilizar para aplicar la firma de la aplicación principal.)*

Para cambiar el certificado:

1. Cree una actualización de la aplicación.
2. Empaquete y firme el archivo de AIR de actualización con el certificado **nuevo**.
3. Vuelva a firmar el archivo de AIR con el certificado **original** (con el comando `-migrate` de ADT).

Un archivo de AIR con firma de migración es, en otros aspectos, un archivo de AIR normal. Si se instala la aplicación en un sistema que no tiene la versión original, AIR la instala de la forma habitual.

Nota: *antes de AIR 1.5.3, la firma de una aplicación de AIR con un certificado renovado no siempre requería una firma de migración. Con el inicio de AIR 1.5.3, una firma de migración siempre es necesaria para los certificados renovados.*

El procedimiento para aplicar una firma de migración se describe en "Firma de un archivo de AIR para cambiar el certificado de la aplicación" en la página 68.

Cambios de la identidad de la aplicación

Antes de AIR 1.5.3, la identidad de una aplicación de AIR cambiaba cuando se instalaba una actualización firmada con una firma de migración. El cambio de identidad de una aplicación tiene varias repercusiones, entre las que se incluyen:

- La nueva versión de la aplicación no tiene acceso a los datos que están en el almacén local cifrado existente.
- Cambia la ubicación del directorio de almacenamiento de la aplicación. Los datos de la ubicación anterior no se copian en el nuevo directorio. (Pero la nueva aplicación puede localizar el directorio original con base en el ID del editor anterior).
- La aplicación ya no puede abrir conexiones locales con el ID del editor anterior.
- La cadena de identidad utilizada para acceder a una aplicación desde una página web cambia.
- El OSID de la aplicación cambia. (El OSID se utiliza al escribir programas de instalación o desinstalación personalizados.)

Al publicar una actualización con AIR 1.5.3, la identidad de la aplicación no puede cambiar. Los ID de editor y la aplicación original se deben especificar en el descriptor de la aplicación del archivo de AIR de actualización. De lo contrario, el nuevo paquete no se reconocerá como actualización.

Nota: *al publicar una nueva aplicación de AIR con AIR 1.5.3 o posterior, no se debe especificar un ID de editor.*

Certificados caducados

A partir de AIR 1.5.3, un certificado que haya caducado en los últimos 180 días aún se puede utilizar para aplicar una firma de migración a una actualización de la aplicación. Para aprovechar este período de gracia, el descriptor de la aplicación de actualización debe especificar 1.5.3 en el atributo de espacio de nombres. Transcurrido el período de gracia, el certificado no se podrá volver a utilizar. Los usuarios que actualicen a una nueva versión de la aplicación tendrán que instalar su versión existente. Se debe tener en cuenta que no existe período de gracia en las versiones de AIR anteriores a 1.5.3.

Terminología

Esta sección contiene un glosario de algunos de los principales términos que necesita conocer al tomar decisiones sobre cómo firmar la aplicación para su distribución pública.

Término	Descripción
Entidad emisora de certificados (AC)	Entidad de una red de infraestructura de claves públicas que interviene como tercero de confianza y que, en última instancia, certifica la identidad del propietario de una clave pública. Una AC emite certificados digitales firmados por su propia clave privada para constatar que ha verificado la identidad del titular del certificado.
Declaración de prácticas de certificación (DPC)	Plantea las prácticas y políticas de la entidad emisora de certificados respecto de la emisión y verificación de certificados. La DPC forma parte del contracto entre la AC y sus suscriptores y partes confiantes. Reseña también las políticas de verificación de identidad y el nivel de garantía que ofrecen los certificados emitidos.
Lista de revocación de certificados (LRC)	Lista de certificados emitidos que han sido revocados y en los cuales ya no se debe confiar. AIR comprueba la LRC en el momento de firmarse una aplicación de AIR y, si no hay marca de hora, nuevamente cuando se instala la aplicación.
Cadena de certificación	Secuencia de certificados en la que cada certificado de la cadena ha sido firmado por el certificado siguiente.
Certificado digital	Documento digital que contiene información acerca de la identidad del propietario, la clave pública del propietario y la identidad del propio certificado. Un certificado emitido por una autoridad de certificación está firmado a su vez por un certificado de propiedad de la AC emisora.
Firma digital	Mensaje o resumen cifrado que sólo puede descifrarse con la clave pública de un par clave pública-clave privada. En una PKI la firma digital contiene un certificado digital (o más) que en última instancia llevan hasta la entidad emisora de certificados. Una firma digital sirve para validar que un mensaje (o un archivo informático) no ha sido alterado desde que se firmó (dentro de los límites de garantía que ofrezca el algoritmo criptográfico que se utilizó) y, suponiendo que se confía en la entidad emisora de certificados, para validar la identidad del firmante.
Almacén de claves	Base de datos que contiene certificados digitales y, en algunos casos, las claves privadas asociadas.
Java Cryptography Architecture (JCA)	Arquitectura de criptografía de Java: arquitectura ampliable para administrar y acceder a los almacenes de claves. Para obtener más información, consulte la guía de consulta de Java Cryptography Architecture (en inglés).
PKCS #11	Cryptographic Token Interface Standard (norma de interfaces de tokens criptográficos) de RSA Laboratories. Un almacén de claves basado en tokens de hardware.
PKCS #12	Personal Information Exchange Syntax Standard (norma para la sintaxis de intercambio de información personal) de RSA Laboratories. Un almacén de claves basado en archivo que suele contener una clave privada y su certificado digital asociado.
Clave privada	La mitad privada de un sistema criptográfico asimétrico que consta de clave pública y clave privada. La clave privada es confidencial y no debe nunca transmitirse a través de una red. Los mensajes con firma digital son cifrados con la clave privada por parte del firmante.

Término	Descripción
Clave pública	La mitad pública de un sistema criptográfico asimétrico que consta de clave pública y clave privada. La clave pública se distribuye libremente y se utiliza para descifrar los mensajes cifrados con la clave privada.
Public Key Infrastructure (PKI)	Infraestructura de claves públicas: sistema de confianza en el que las autoridades de certificación autentican la identidad de los propietarios de claves públicas. Los clientes de la red confían en los certificados digitales emitidos por una AC de confianza para verificar la identidad del firmante de un mensaje o archivo digital.
Marca de hora	Dato firmando digitalmente que contiene la fecha y hora en que se produjo un evento. ADT puede incluir en un paquete de AIR una marca de hora provista por un servidor de hora en conformidad con la norma RFC 3161. Cuando la hay, AIR utiliza la marca de hora para establecer la validez de un certificado en el momento de firmar. Esto permite instalar una aplicación de AIR una vez caducado su certificado de firma.
Autoridad de marcas de hora	Autoridad que emite marcas de hora. Para que AIR la reconozca, la marca de hora debe estar en conformidad con la norma RFC 3161 y la firma de la marca de hora debe encadenarse con un certificado raíz de confianza en la máquina en que se instale la aplicación.

Capítulo 14: Actualización de aplicaciones de AIR

Los usuarios pueden instalar o actualizar cualquier aplicación de AIR haciendo doble clic en el archivo de AIR de su equipo o desde un navegador (mediante la perfeccionada función de instalación). El instalador de Adobe® AIR™ gestiona la instalación y avisa al usuario si está actualizando una aplicación previa existente. (Consulte "Distribución, instalación y ejecución de aplicaciones de AIR" en la página 86.)

Sin embargo, también es posible permitir que las propias aplicaciones se actualicen solas mediante la clase Updater. (Una aplicación instalada puede detectar nuevas versiones disponibles para su descarga e instalación.) La clase Updater incluye un método `update()` que permite al usuario apuntar a un archivo de AIR de un equipo y actualizar a dicha versión.

Tanto el ID de aplicación como el ID de editor de un archivo de actualización de AIR deben coincidir para que la aplicación se actualice. El ID de editor proviene del certificado de firma. Tanto la actualización como la aplicación que va a actualizarse deben estar firmadas con el mismo certificado.

Para AIR 1.5.3 o posterior, el archivo descriptor de la aplicación incluye un elemento `<publisherID>`. Este elemento debe usarse si existen versiones de la aplicación desarrolladas utilizando AIR 1.5.2 o una versión anterior. Para obtener más información, consulte "Definición de la identidad de la aplicación" en la página 74.

En AIR 1.1 y posterior, es posible migrar una aplicación para utilizar un nuevo certificado de firma para el código. Para migrar una aplicación y utilizar una nueva firma, es preciso firmar el archivo de actualización de AIR con el certificado nuevo y con el original. La migración de certificados es un proceso que no se puede invertir. Una vez concluida la migración, sólo se reconocerán como actualizaciones de la instalación existente aquellos archivos de AIR firmados con el nuevo certificado (o con ambos certificados).

La administración de las actualizaciones de aplicaciones puede resultar un proceso complicado. AIR 1.5 incluye el nuevo *marco de actualización para las aplicaciones de AdobeAIR*. Este marco proporciona las API que ayudan a los desarrolladores a ofrecer buenas capacidades de actualización en aplicaciones de AIR.

Puede utilizar la migración de certificados para pasar de un certificado firmado automáticamente a un certificado comercial de firma de código, o de uno firmado automáticamente a otro del mismo tipo. Si no migra el certificado, los usuarios existentes deberán quitar su versión actual de la aplicación para poder instalar la nueva versión. Para obtener más información, consulte "Cambio de certificado" en la página 101.

Es recomendable incluir un mecanismo de actualización en la aplicación. Si se crea una nueva versión de la aplicación, el mecanismo de actualización puede indicar al usuario que instale la nueva versión.

El instalador de aplicaciones de AIR crea archivos de registro cuando se instala, se actualiza o se elimina una aplicación de AIR. Puede consultar estos registros para ayudar a determinar la causa de cualquier problema de instalación. Consulte "Registros de instalación" en la página 96.

Nota: *las nuevas versiones del motor de ejecución Adobe AIR puede incluir versiones actualizadas de WebKit. Una versión actualizada de WebKit puede implicar cambios inesperados en el contenido HTML de una aplicación implementada de AIR. Estos cambios pueden requerir la actualización de la aplicación. Un mecanismo de actualización puede informar al usuario de la nueva versión de la aplicación. Para obtener más información, consulte Entorno HTML (para desarrolladores de ActionScript) o Entorno HTML (para desarrolladores HTML).*

Actualización de aplicaciones

La clase Updater (del paquete flash.desktop) incluye un método, `update()`, que se puede utilizar para actualizar la aplicación actualmente en ejecución a una versión distinta. Por ejemplo, si el usuario tiene una versión del archivo de AIR ("Sample_App_v2.air") en el escritorio, el siguiente código actualizaría la aplicación:

Ejemplo de ActionScript:

```
var updater:Updater = new Updater();
var airFile:File = File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version:String = "2.01";
updater.update(airFile, version);
```

Ejemplo de JavaScript:

```
var updater = new air.Updater();
var airFile = air.File.desktopDirectory.resolvePath("Sample_App_v2.air");
var version = "2.01";
updater.update(airFile, version);
```

Antes de que una aplicación utilice la clase Updater, el usuario o la aplicación deben descargar la versión actualizada del archivo de AIR en el equipo. Para obtener más información, consulte "Descarga de un archivo de AIR en el equipo del usuario" en la página 108.

Resultados de la llamada al método Updater.update()

Cuando una aplicación del motor de ejecución llama al método `update()`, éste cierra la aplicación y, a continuación, intenta instalar la nueva versión del archivo de AIR. Se comprueba que el ID de aplicación y el ID de editor especificados en el archivo de AIR coinciden con el ID de aplicación y de editor de la aplicación que llama al método `update()`. (Para obtener más información sobre el ID de aplicación y el ID de editor, consulte "Configuración de las propiedades de una aplicación de AIR" en la página 71.) También comprueba si la cadena de versión coincide con la cadena `version` transferida al método `update()`. Si la instalación concluye correctamente, el motor de ejecución abre la nueva versión de la aplicación. En caso contrario (si la instalación no concluye correctamente), vuelve a abrir la versión existente de la aplicación (previa a la instalación).

En Mac OS, para instalar una versión actualizada de una aplicación, el usuario debe contar con adecuados privilegios del sistema para instalar en el directorio de la aplicación. En Windows y Linux, el usuario debe disponer de privilegios de administrador.

Si la versión actualizada de la aplicación requiere una versión actualizada del motor de ejecución, se instala la versión más reciente del motor de ejecución. Para actualizar el motor de ejecución, el usuario debe tener privilegios administrativos para el equipo.

Al verificar una aplicación con ADL, llamar al método `update()` produce una excepción de tiempo de ejecución.

Cadena de versión

Para que el archivo de AIR se pueda instalar, la cadena que se especifica como parámetro `version` del método `update()` debe coincidir con la cadena del atributo `version` del elemento principal `application` del archivo descriptor de la aplicación. Es preciso especificar el parámetro `version` por motivos de seguridad. Al solicitar a la aplicación que verifique el número de versión en el archivo de AIR, la aplicación no instalará por error una versión anterior. (Una versión anterior de la aplicación puede presentar una vulnerabilidad de seguridad que se haya solucionado en la aplicación instalada actualmente.) La aplicación también comprueba la cadena de versión en el archivo de AIR y la compara con la de la aplicación instalada para evitar desactualizaciones.

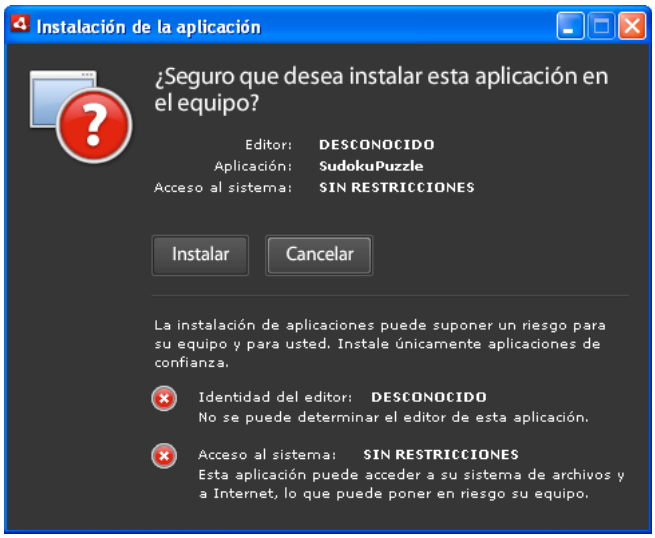
La cadena de versión puede tener cualquier formato. Por ejemplo, "2.01" o "versión 2". Es el usuario (el desarrollador de la aplicación) quien debe decidir qué formato darle a esta cadena. El motor de ejecución no valida la cadena de versión; el código de la aplicación debe hacerlo antes de actualizar la aplicación.

Si una aplicación de Adobe AIR descarga un archivo de AIR por Internet, se recomienda disponer de un mecanismo que permite al servicio Web notificar a la aplicación de Adobe AIR sobre la descarga actual de la versión. La aplicación puede utilizar esta cadena como el parámetro `version` del método `update()`. Si el archivo de AIR se obtiene por otros medios que impiden conocer la versión del archivo de AIR, la aplicación de AIR puede examinar el archivo de AIR para extraer la información sobre su versión. (Un archivo de AIR es un archivo ZIP comprimido y el archivo descriptor de la aplicación es el segundo registro del archivo.)

Para obtener más información sobre el archivo descriptor de aplicación, consulte “Configuración de las propiedades de una aplicación de AIR” en la página 71.

Presentación de una interfaz de usuario personalizada para las actualizaciones de la aplicación

AIR incluye una interfaz de actualización predeterminada:



Esta interfaz siempre se utiliza la primera vez que el usuario instala la versión de una aplicación en un ordenador. Sin embargo, es posible definir una interfaz propia para utilizarla en el futuro. Si su aplicación define una interfaz de actualización personalizada, especifique un elemento `customUpdateUI` en el archivo descriptor de la aplicación para la aplicación instalada actualmente:

```
<customUpdateUI>true</customUpdateUI>
```

Cuando la aplicación se instale y el usuario abra un archivo de AIR con un ID de aplicación e ID de editor que coincidan con los de la aplicación instalada, será el motor de ejecución el encargado de abrir la aplicación, no el archivo de instalación predeterminado de la aplicación de AIR. Para obtener más información, consulte “Interfaz de usuario personalizada para actualizaciones” en la página 80.

La aplicación puede decidir, cuando se ejecute (cuando el objeto `NativeApplication.nativeApplication` distribuye un evento `load`), si actualiza o no la aplicación (con la clase Updater). Si decide actualizarse, puede presentar su propia interfaz de instalación al usuario (interfaz que no es igual que la estándar).

Descarga de un archivo de AIR en el equipo del usuario

Para poder utilizar la clase Updater, el usuario o la aplicación deben guardar primero localmente un archivo de AIR en el equipo del usuario.

Nota: *AIR 1.5 incluye un marco de actualización que ayuda a los desarrolladores a ofrecer buenas capacidades de actualización en aplicaciones de AIR. El uso de este marco puede resultar mucho más sencillo que la utilización del método* `update()` *de la clase Update directamente. Para obtener más información, consulte "Utilización del marco de actualización" en la página 112.*

El siguiente código lee un archivo de AIR desde una dirección URL (http://example.com/air/updates/Sample_App_v2.air) y lo guarda en el directorio de almacenamiento de la aplicación:

Ejemplo de ActionScript:

```
var urlString:String = "http://example.com/air/updates/Sample_App_v2.air";
var urlReq:URLRequest = new URLRequest(urlString);
var urlStream:URLStream = new URLStream();
var fileData:ByteArray = new ByteArray();
urlStream.addEventListener(Event.COMPLETE, loaded);
urlStream.load(urlReq);

function loaded(event:Event):void {
    urlStream.readBytes(fileData, 0, urlStream.bytesAvailable);
    writeAirFile();
}

function writeAirFile():void {
    var file:File = File.applicationStorageDirectory.resolvePath("My App v2.air");
    var fileStream:FileStream = new FileStream();
    fileStream.open(file, FileMode.WRITE);
    fileStream.writeBytes(fileData, 0, fileData.length);
    fileStream.close();
    trace("The AIR file is written.");
}
```

Ejemplo de JavaScript:

```
var urlString = "http://example.com/air/updates/Sample_App_v2.air";
var urlReq = new air.URLRequest(urlString);
var urlStream = new air.URLStream();
var fileData = new air.ByteArray();
urlStream.addEventListener(air.Event.COMPLETE, loaded);
urlStream.load(urlReq);

function loaded(event) {
    urlStream.readBytes(fileData, 0, urlStream.bytesAvailable);
    writeAirFile();
}

function writeAirFile() {
    var file = air.File.desktopDirectory.resolvePath("My App v2.air");
    var fileStream = new air.FileStream();
    fileStream.open(file, air.FileMode.WRITE);
    fileStream.writeBytes(fileData, 0, fileData.length);
    fileStream.close();
    trace("The AIR file is written.");
}
```

Para obtener más información, consulte:

- Flujo de trabajo de lectura y escritura de archivos (para desarrolladores de ActionScript)
- Flujo de trabajo de lectura y escritura de archivos (para desarrolladores de HTML)

Comprobar si una aplicación se está ejecutando por primera vez

Una vez actualizada la aplicación, puede ofrecer al usuario un mensaje de bienvenida o de primeros pasos. Al iniciarse, la aplicación comprueba si se está ejecutando por primera vez para saber si debe mostrar o no el mensaje.

Nota: *AIR 1.5 incluye un marco de actualización que ayuda a los desarrolladores a ofrecer buenas capacidades de actualización en aplicaciones de AIR. Este marco proporciona métodos sencillos para comprobar si una versión de una aplicación se está ejecutando por primera vez. Para obtener más información, consulte "Utilización del marco de actualización" en la página 112.*

Una forma de hacerlo es guardar un archivo en el directorio de almacenamiento de la aplicación al inicializarla. Cada vez que se inicie la aplicación, comprobará si existe este archivo. Si el archivo no existe, significa que la aplicación se está ejecutando por primera vez para el usuario. Si el archivo existe, significa que el usuario ya ha ejecutado la aplicación al menos una vez. Si el archivo existe y su número de versión es anterior a la versión actual, significa que el usuario está ejecutando la aplicación por primera vez.

Este concepto se demuestra en el siguiente ejemplo de Flex:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:WindowedApplication xmlns:mx="http://www.adobe.com/2006/mxml"
    layout="vertical"
    title="Sample Version Checker Application"
    applicationComplete="system extension()">
    <mx:Script>
        <![CDATA[
            import flash.filesystem.*;
            public var file:File;
            public var currentVersion:String = "1.2";
            public function system extension():void {
                file = File.applicationStorageDirectory;
                file = file.resolvePath("Preferences/version.txt");
                trace(file.nativePath);
                if(file.exists) {
                    checkVersion();
                } else {
                    firstRun();
                }
            }
            private function checkVersion():void {
                var stream:FileStream = new FileStream();
                stream.open(file, FileMode.READ);
                var reversion:String = stream.readUTFBytes(stream.bytesAvailable);
                stream.close();
                if (reversion != currentVersion) {
                    log.text = "You have updated to version " + currentVersion + ".\n";
                } else {
                    saveFile();
                }
                log.text += "Welcome to the application.";
            }
            private function firstRun():void {
                log.text = "Thank you for installing the application. \n"
                    + "This is the first time you have run it.";
                saveFile();
            }
            private function saveFile():void {
                var stream:FileStream = new FileStream();
                stream.open(file, FileMode.WRITE);
                stream.writeUTFBytes(currentVersion);
                stream.close();
            }
        ]]>
    </mx:Script>
    <mx:TextArea ID="log" width="100%" height="100%" />
</mx:WindowedApplication>
```

Este concepto se muestra en el siguiente ejemplo de JavaScript:

```
<html>
    <head>
        <script src="AIRAliases.js" />
        <script>
            var file;
            var currentVersion = "1.2";
            function system extension() {
                file = air.File.appStorageDirectory.resolvePath("Preferences/version.txt");
                air.trace(file.nativePath);
                if(file.exists) {
                    checkVersion();
                } else {
                    firstRun();
                }
            }
            function checkVersion() {
                var stream = new air.FileStream();
                stream.open(file, air.FileMode.READ);
                var reversion = stream.readUTFBytes(stream.bytesAvailable);
                stream.close();
                if (reversion != currentVersion) {
                    window.document.getElementById("log").innerHTML
                            = "You have updated to version " + currentVersion + ".\n";
                } else {
                    saveFile();
                }
                window.document.getElementById("log").innerHTML
                                += "Welcome to the application.";
            }
            function firstRun() {
                window.document.getElementById("log").innerHTML
                            = "Thank you for installing the application. \n"
                            + "This is the first time you have run it.";
                saveFile();
            }
            function saveFile() {
                var stream = new air.FileStream();
                stream.open(file, air.FileMode.WRITE);
                stream.writeUTFBytes(currentVersion);
                stream.close();
            }
        </script>
    </head>
    <body onLoad="system extension()">
        <textarea ID="log" rows="100%" cols="100%" />
    </body>
</html>
```

Si la aplicación guarda datos localmente (por ejemplo, en el directorio de almacenamiento de la aplicación), puede buscar datos guardados previamente (de versiones anteriores) durante la primera ejecución de la aplicación.

Utilización del marco de actualización

La administración de las actualizaciones de aplicaciones puede resultar un proceso complicado. El *marco de actualización para las aplicaciones de AdobeAIR*proporciona las API que ayudan a los desarrolladores a ofrecer buenas capacidades de actualización en aplicaciones de AIR. La funcionalidad del marco de actualización de AIR ayuda a los desarrolladores del siguiente modo:

- Búsqueda periódica de actualizaciones en función de un intervalo o cuando el usuario lo solicita.
- Descarga de archivos AIR (actualizaciones) desde un origen Web.
- Aviso al usuario de la primera ejecución de la versión recién instalada.
- Confirmación de que el usuario desea buscar actualizaciones.
- Información para el usuario de la nueva versión de actualización.
- Información para el usuario sobre el progreso de la descarga y errores.

El marco de actualización de AIR proporciona una interfaz de usuario de ejemplo que puede utilizar su aplicación. Ofrece al usuario información básica y opciones relacionadas con las actualizaciones de la aplicación. La aplicación también puede definir su propia interfaz de usuario para su uso con el marco de actualización.

El marco de actualización de AIR permite almacenar información sobre la versión de actualización de una aplicación de AIR en sencillos archivos de configuración XML. Para la mayoría de las aplicaciones, la definición de estos archivos de configuración y la adición de código básico proporciona una buena funcionalidad de actualización para el usuario final.

Aun sin el uso del marco de actualización, Adobe AIR incluye una clase Updater que las aplicaciones de AIR pueden emplear para actualizar a nuevas versiones. Esta clase permite que una aplicación se actualice a una versión incluida en un archivo de AIR en el equipo del usuario. No obstante, la administración de la actualización puede implicar un proceso más complejo que el basar simplemente la actualización de la aplicación en un archivo de AIR almacenado localmente.

Archivos en el marco de actualización de AIR

El marco de actualización de AIR se incluye en el directorio frameworks/libs/air del SDK de AIR 2. Incluye los siguientes archivos:

- applicationupdater.swc: define la funcionalidad básica de la biblioteca de actualización, para su uso en ActionScript. Esta versión no contiene interfaz de usuario.
- applicationupdater.swf: define la funcionalidad básica de la biblioteca de actualización, para su uso en JavaScript. Esta versión no contiene interfaz de usuario.
- applicationupdater_ui.swc: define la funcionalidad básica de la biblioteca de actualización de la versión Flex 4, incluyendo una interfaz de usuario que la aplicación puede utilizar para mostrar las opciones de actualización.
- applicationupdater_ui.swf: define la funcionalidad básica de la biblioteca de actualización de la versión de JavaScript, incluyendo una interfaz de usuario que la aplicación puede utilizar para mostrar las opciones de actualización.
- flex3/applicationupdater_ui.swc: la versión del archivo applicationupdater_ui para su uso con Flex 3.

  Importante: si se utiliza el SDK de AIR 2 con Flex 3, asegúrese de hacer referencia a la versión del archivo applicationupdater_ui.swc en el subdirectorio flex3. (O bien, puede reemplazar la versión en el directorio frameworks/libs/air por la versión del subdirectorio flex3.)

Para obtener más información, consulte las siguientes secciones:

Configuración del entorno de desarrollo de Flex

Los archivos SWC del directorio frameworks/libs/air del SDK de AIR 2 definen clases que se pueden utilizar en el desarrollo de Flex y Flash.

Para utilizar el marco de actualización al compilar con el SDK de Flex, incluya el archivo ApplicationUpdater.swc o ApplicationUpdater_UI.swc en la llamada al compilador amxmlc. En el siguiente ejemplo, el compilador carga el archivo ApplicationUpdater.swc en el subdirectorio lib del directorio Flex SDK:

```
amxmlc -library-path+=lib/ApplicationUpdater.swc  -- myApp.mxml
```

En el siguiente ejemplo, el compilador carga el archivo ApplicationUpdater_UI.swc en el subdirectorio lib del directorio Flex SDK:

```
amxmlc -library-path+=lib/ApplicationUpdater_UI.swc  -- myApp.mxml
```

Al realizar el desarrollo con el uso de Flash Builder, añada el archivo SWC en la ficha de ruta de biblioteca de la configuración de la ruta de compilación de Flex en el cuadro de diálogo de propiedades.

Asegúrese de copiar los archivos SWC en el directorio al que se hará referencia en el compilador amxmlc (usando el SDK de Flex) o Flash Builder.

Inclusión de archivos del marco en una aplicación de AIR basada en HTML

El directorio frameworks/html del marco de actualización incluye los siguientes archivos SWF:

- ApplicationUpdater.swf: define la funcionalidad básica de la biblioteca de actualización, sin ninguna interfaz de usuario.
- ApplicationUpdater_UI.swf: define la funcionalidad básica de la biblioteca de actualización, incluyendo una interfaz de usuario que la aplicación puede utilizar para mostrar las opciones de actualización.

El código JavaScript de las aplicaciones de AIR puede utilizar clases definidas en archivos SWF.

Para utilizar el marco de actualización, incluya el archivo ApplicationUpdater.swf o ApplicationUpdater_UI.swf en el directorio de la aplicación (o un subdirectorio). A continuación, en el archivo HTML que utilizará el marco (en código JavaScript), incluya una etiqueta `script` que cargue el archivo:

```
<script src="applicationUpdater.swf" type="application/x-shockwave-flash"/>
```

O bien, utilice esta etiqueta `script` para cargar el archivo ApplicationUpdater_UI.swf:

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

La API definida en estos dos archivos se describe en el resto del documento.

Ejemplo básico: Uso de la versión ApplicationUpdaterUI

La versión ApplicationUpdaterUI del marco de actualización proporciona una interfaz básica que se puede emplear fácilmente en la aplicación. A continuación se incluye un ejemplo básico.

En primer lugar, cree una aplicación de AIR que llame al marco de actualización:

1 Si su aplicación es una aplicación de AIR basada en HTML, cargue el archivo ApplicationUpdaterUI.js:

```
<script src="ApplicationUpdater_UI.swf" type="application/x-shockwave-flash"/>
```

2 En la lógica del programa de su aplicación de AIR, cree una instancia de un objeto ApplicationUpdaterUI.

En ActionScript, utilice el siguiente código:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

En JavaScript, utilice el siguiente código:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

Es posible que desee añadir este código en una función de inicialización que se ejecute una vez cargada la aplicación.

3 Cree un archivo de texto denominado updateConfig.xml y añádale lo siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
    <url>http://example.com/updates/update.xml</url>
    <delay>1</delay>
</configuration>
```

Edite el elemento `URL` del archivo updateConfig.xml para que coincida con la ubicación final del archivo descriptor de actualización en su servidor web (consulte el siguiente procedimiento).

`delay` es el número de días que la aplicación espera entre las búsquedas de actualizaciones.

4 Añada el archivo updateConfig.xml al directorio del proyecto de su aplicación de AIR.

5 Haga que el objeto updater haga referencia al archivo updateConfig.xml y llame al método `initialize()` del objeto.

En ActionScript, utilice el siguiente código:

```
appUpdater.configurationFile = new File("app:/updateConfig.xml");
appUpdater.initialize();
```

En JavaScript, utilice el siguiente código:

```
appUpdater.configurationFile = new air.File("app:/updateConfig.xml");
appUpdater.initialize();
```

6 Cree una segunda versión de la aplicación de AIR que tenga una versión distinta a la primera aplicación. (La versión se especifica en el archivo descriptor de la aplicación, en el elemento `version`.)

A continuación, añada la versión de actualización de la aplicación de AIR al servidor web:

1 Sitúe la versión de actualización del archivo de AIR en el servidor web.

2 Cree un archivo de texto denominado updateDescriptor.xml y agréguele el siguiente contenido:

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1</version>
       <url>http://example.com/updates/sample_1.1.air</url>
       <description>This is the latest version of the Sample application.</description>
    </update>
```

Edite los elementos `version`, `URL` y `description` del archivo updateDescriptor.xml para que coincida con el archivo de AIR de actualización.

3 Añada el archivo updateDescriptor.xml al mismo directorio del servidor web que contiene el archivo de AIR de actualización.

Este es un ejemplo básico, pero proporciona la funcionalidad de actualización suficiente para diversas aplicaciones. En el resto del documento se describe cómo emplear el marco de actualización para que mejor se adapte a sus necesidades.

Para obtener otro ejemplo del uso de marco de actualización, consulte las siguientes aplicaciones de ejemplo en el entro de desarrollo de Adobe AIR:

- Update Framework in a Flex-based Application (http://www.adobe.com/go/learn_air_qs_update_framework_flex_en) (Marco de actualización en una aplicación basada en Flex; en inglés)
- Update Framework in a Flash-based Application (Marco de actualización en una aplicación basada en Flash; en inglés)(http://www.adobe.com/go/learn_air_qs_update_framework_flash_en)
- Update Framework in a HTML-based Application (http://www.adobe.com/go/learn_air_qs_update_framework_html_en) (Marco de actualización en una aplicación basada en HTML; en inglés)

Definición del archivo descriptor de actualización y adición del archivo de AIR al servidor web

Cuando se utiliza el marco de actualización de AIR, se define la información básica sobre la actualización disponible en un archivo descriptor de actualización, almacenado en el servidor web. El archivo descriptor de actualización es un sencillo archivo XML. El marco de actualización incluido en la aplicación comprueba este archivo para ver si se ha cargado una nueva versión.

El archivo descriptor de actualización contiene los siguientes datos:

- `version`: la nueva versión de la aplicación de AIR. Debe ser la misma cadena que se usa en el nuevo archivo descriptor de la aplicación de AIR como versión. Si la versión del archivo descriptor no coincide con la versión del archivo de actualización de AIR, el marco de actualización emitirá una excepción.
- `url`: ubicación del archivo de actualización de AIR. Se trata del archivo que contiene la versión de actualización de la aplicación de AIR.
- `description`: información sobre la nueva versión. Esta información se puede mostrar al usuario durante el proceso de actualización.

Los elementos `version` y `url` son obligatorios. El elemento `description` es opcional.

A continuación se incluye un archivo descriptor de actualización de ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1a1</version>
       <url>http://example.com/updates/sample_1.1a1.air</url>
       <description>This is the latest version of the Sample application.</description>
    </update>
```

Si desea definir la etiqueta `description` mediante varios idiomas, utilice varios elementos `text` que definan un atributo `lang`:

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1a1</version>
       <url>http://example.com/updates/sample_1.1a1.air</url>
       <description>
           <text xml:lang="en">English description</text>
           <text xml:lang="fr">French description</text>
           <text xml:lang="ro">Romanian description</text>
       </description>
    </update>
```

Sitúe el archivo descriptor de actualización, junto con el archivo de actualización de AIR, en el servidor web.

El directorio de plantillas incluido con el descriptor de actualización contiene archivos descriptores de actualización de ejemplo. Éstos incluyen versiones en un solo idioma y varios idiomas.

Creación de una instancia del objeto updater

Tras cargar el marco de actualización de AIR en su código (consulte "Configuración del entorno de desarrollo de Flex" en la página 113 e "Inclusión de archivos del marco en una aplicación de AIR basada en HTML" en la página 113), es necesario crear una instancia del objeto updater, tal y como se muestra en el siguiente ejemplo.

Ejemplo de ActionScript:

```
var appUpdater:ApplicationUpdater = new ApplicationUpdater();
```

Ejemplo de JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdater();
```

El código anterior utiliza la clase ApplicationUpdater (que no proporciona ninguna interfaz de usuario). Si desea usar la clase ApplicationUpdaterUI (que proporciona una interfaz de usuario), utilice lo siguiente:

Ejemplo de ActionScript:

```
var appUpdater:ApplicationUpdaterUI = new ApplicationUpdaterUI();
```

Ejemplo de JavaScript:

```
var appUpdater = new runtime.air.update.ApplicationUpdaterUI();
```

En los ejemplos de código restantes de este documento se presupone que se ha creado una instancia de un objeto updater denominada `appUpdater`.

Definición de la configuración de actualización

Tanto ApplicationUpdater como ApplicationUpdaterUI se pueden configurar mediante un archivo de configuración incluido con la aplicación, o bien, a través de ActionScript o JavaScript en la aplicación.

Definición de la configuración de actualización en un archivo de configuración XML

El archivo de configuración de actualización es un archivo XML. Puede incluir los siguientes elementos:

- `updateURL`: una cadena. Representa la ubicación del descriptor de actualización en el servidor remoto. Se permite cualquier ubicación URLRequest válida. Se debe definir la propiedad `updateURL`, a través del archivo de configuración o mediante un script (consulte "Definición del archivo descriptor de actualización y adición del archivo de AIR al servidor web" en la página 115). Esta propiedad se debe definir antes de utilizar el elemento updater (antes de llamar al método `initialize()` del objeto updater, descrito en "Inicialización del marco de actualización" en la página 119).

- `delay`: un número. Representa un intervalo de tiempo determinado en días (se permiten valores similares a `0,25`) para buscar actualizaciones. El valor de 0 (valor predeterminado) especifica que updater no realiza ninguna comprobación periódica automática.

El archivo de configuración para ApplicationUpdaterUI puede incluir el siguiente elemento además de los elementos `updateURL` y `delay`:

- `defaultUI`: una lista de elementos`dialog`. Cada elemento `dialog` dispone de un atributo `name` que se corresponde con el cuadro de diálogo en la interfaz de usuario. Todos los elementos `dialog` cuentan con un atributo `visible` que define si el cuadro de diálogo es visible. El valor predeterminado es `true`. Entre los posibles valores para el atributo `name` se encuentran los siguientes:
  - `"checkForUpdate"`: corresponde a los cuadros de diálogo de búsqueda de actualizaciones, sin actualizaciones y error de actualización.
  - `"downloadUpdate"`: corresponde al cuadro de diálogo de descarga de actualizaciones.
  - `"downloadProgress"`: corresponde a los cuadros de diálogo de progreso y error de descarga.
  - `"installUpdate"`: corresponde al cuadro de diálogo de instalación de la actualización.
  - `"fileUpdate"`: corresponde a los cuadros de diálogo de actualización de archivo, no actualización de archivo y error de archivo.
- `"unexpectedError"`: corresponde al cuadro de diálogo de error inesperado.

  Cuando se establece en `false`, el cuadro de diálogo correspondiente no aparece como parte del proceso de actualización.

A continuación se incluye un ejemplo del archivo de configuración para el marco ApplicationUpdater:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
      <url>http://example.com/updates/update.xml</url>
      <delay>1</delay>
</configuration>
```

Se muestra un ejemplo del archivo de configuración para el marco ApplicationUpdaterUI, que incluye una definición para el elemento `defaultUI`:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns="http://ns.adobe.com/air/framework/update/configuration/1.0">
      <url>http://example.com/updates/update.xml</url>
      <delay>1</delay>
      <defaultUI>
         <dialog name="checkForUpdate" visible="false" />
         <dialog name="downloadUpdate" visible="false" />
         <dialog name="downloadProgress" visible="false" />
      </defaultUI>
</configuration>
```

Seleccione la propiedad `configurationFile` en la ubicación de ese archivo:

Ejemplo de ActionScript:

```
appUpdater.configurationFile = new File("app:/cfg/updateConfig.xml");
```

Ejemplo de JavaScript:

```
appUpdater.configurationFile = new air.File("app:/cfg/updateConfig.xml");
```

El directorio de plantillas del marco de actualización incluye un archivo de configuración de ejemplo, config-template.xml.

Definición de la configuración de actualización con código ActionScript o JavaScript

Estos parámetros de configuración también se pueden establecer utilizando código en la aplicación, tal y como se muestra a continuación:

```
appUpdater.updateURL = " http://example.com/updates/update.xml";
appUpdater.delay = 1;
```

Las propiedades del objeto updater son `updateURL` y `delay`. Estas propiedades definen la misma configuración que los elementos `updateURL` y `delay` en el archivo de configuración: la dirección URL del archivo descriptor de actualización y el intervalo para buscar actualizaciones. Si se especifica un archivo de configuración *y* la configuración en código, todas las propiedades establecidas utilizando código tienen prioridad sobre la configuración correspondiente en el archivo de configuración.

Se debe definir la propiedad `updateURL`, a través del archivo de configuración o mediante script (consulte "Definición del archivo descriptor de actualización y adición del archivo de AIR al servidor web" en la página 115) antes de utilizar updater (antes de llamar al método `initialize()` del objeto updater, descrito en "Inicialización del marco de actualización" en la página 119).

El marco ApplicationUpdaterUI define estas propiedades adicionales del objeto updater:

- `isCheckForUpdateVisible`: corresponde a los cuadros de diálogo de búsqueda de actualizaciones, sin actualizaciones y error de actualización.
- `isDownloadUpdateVisible`: corresponde al cuadro de diálogo de descarga de actualizaciones.
- `isDownloadProgressVisible`: corresponde a los cuadros de diálogo de progreso y error de descarga.
- `isInstallUpdateVisible`: corresponde al cuadro de diálogo de instalación de la actualización.
- `isFileUpdateVisible`: corresponde a los cuadros de diálogo de actualización de archivo, no actualización de archivo y error de archivo.
- `isUnexpectedErrorVisible`: corresponde al cuadro de diálogo de error inesperado.

Todas las propiedades hacen referencia a uno o varios cuadros de diálogo en la interfaz de usuario ApplicationUpdaterUI. Cada propiedad es un valor booleano, con un valor predeterminado de `true`. Cuando se establece en `false`, los cuadros de diálogo correspondientes no aparecen como parte del proceso de actualización.

Estas propiedades de cuadro de diálogo anulan la configuración del archivo de configuración de actualización.

Proceso de actualización

El marco de actualización de AIR completa el proceso de actualización en los siguientes pasos:

1. Con la inicialización de updater se verifica si se ha realizado una comprobación de actualización en el intervalo de días definido (consulte "Definición de la configuración de actualización" en la página 116). Si hay programada una comprobación de actualización, el proceso de actualización continúa.
2. Updater descarga e interpreta el archivo descriptor de actualización.
3. También se descarga el archivo de actualización de AIR.
4. Updater instala la versión actualizada de la aplicación.

El objeto updater distribuye eventos a la finalización de cada uno de estos pasos. En la versión de ApplicationUpdater, es posible cancelar los eventos que indiquen la correcta finalización de un paso en el proceso. Si cancela uno de estos eventos, se cancelará el siguiente paso del proceso. En la versión de ApplicationUpdaterUI, updater presenta un cuadro de diálogo que permite al usuario cancelar o continuar en cada paso del proceso.

Si cancela el evento, se pueden llamar a métodos del objeto updater para reanudar el proceso.

Conforme progresa la versión ApplicationUpdater de updater en el proceso de actualización, se registra su estado actual, en una propiedad `currentState`. Esta propiedad se establece en una cadena los siguientes posibles valores:

- `"UNINITIALIZED"`: updater no se ha inicializado.
- `"INITIALIZING"`: updater se está inicializando.
- `"READY"`: updater se ha inicializado.
- `"BEFORE_CHECKING"`: updater no ha comprobado aún el archivo descriptor de actualización.
- `"CHECKING"`: updater está buscando un archivo descriptor de actualización.
- `"AVAILABLE"`: el archivo descriptor de updater está disponible.
- `"DOWNLOADING"`: updater está descargando el archivo de AIR.
- `"DOWNLOADED"`: updater ha descargado el archivo de AIR.
- `"INSTALLING"`: updater está instalando el archivo de AIR.
- `"PENDING_INSTALLING"`: updater se ha inicializado y hay actualizaciones pendientes.

Algunos métodos del objeto updater sólo se ejecutan si updater se encuentra en un determinado estado.

Inicialización del marco de actualización

Una vez definidas las propiedades de configuración, (consulte "Ejemplo básico: Uso de la versión ApplicationUpdaterUI" en la página 113), llame al método `initialize()` para inicializar la actualización:

```
appUpdater.initialize();
```

Este método realiza lo siguiente:

- Inicializa el marco de actualización, realizando una instalación silenciosa de forma sincrónica de todas las actualizaciones pendientes. Es necesario llamar a este método durante el inicio de la aplicación, ya que es posible que reinicie la aplicación cuando se llame.
- Compruebe si hay alguna actualización pendiente y la instala.
- Si se produce un error durante el proceso de actualización, borra el archivo de actualización y la información de la versión del área de almacenamiento de la aplicación.
- Si el intervalo de días de comprobación de actualización ha caducado, inicia el proceso de actualización. De lo contrario, reinicia el temporizador.

La llamada a este método puede provocar que el objeto updater distribuya los siguientes eventos:

- `UpdateEvent.INITIALIZED`: se distribuye cuando se completa la inicialización.
- `ErrorEvent.ERROR`: se distribuye cuando se produce un error durante la inicialización.

Al distribuir el evento `UpdateEvent.INITIALIZED`, el proceso de actualización se completa.

Cuando se llamada al método `initialize()`, updater inicia el proceso de actualización y completa todos los pasos, en función de la configuración del intervalo de demora del temporizador. No obstante, también puede iniciar el proceso de actualización en cualquier momento llamando al método `checkNow()` del objeto updater:

```
appUpdater.checkNow();
```

Este método no realiza ninguna operación si el proceso de actualización ya se está ejecutando. De lo contrario, inicia el proceso de actualización.

El objeto updater puede distribuir el siguiente evento como resultado de la llamada al método `checkNow()`:

- `UpdateEvent.CHECK_FOR_UPDATE` justo antes de que intente descargar el archivo descriptor de actualización.

Si se cancela el evento `checkForUpdate`, se puede llamar al método `checkForUpdate()` del objeto updater. (Consulte la siguiente sección.) Si no cancela el evento, el proceso de actualización continúa comprobando el archivo descriptor de actualización.

Administración del proceso de actualización en la versión de ApplicationUpdaterUI

En la versión de ApplicationUpdaterUI, el usuario puede cancelar el proceso mediante los botones Cancelar de los cuadros de diálogo de la interfaz de usuario. Asimismo, es posible cancelar mediante programación el proceso de actualización llamando al método `cancelUpdate()` del objeto ApplicationUpdaterUI.

Se pueden establecer las propiedades del objeto ApplicationUpdaterUI o definir elementos en el archivo de configuración de actualización para especificar qué confirmaciones de cuadro de diálogo muestra updater. Para obtener más información, consulte "Definición de la configuración de actualización" en la página 116.

Administración del proceso de actualización en la versión de ApplicationUpdater

Puede llamara al método `preventDefault()` de los objetos de evento distribuidos mediante el objeto ApplicationUpdater con el fin de cancelar pasos del proceso de actualización (consulte "Proceso de actualización" en la página 118). La cancelación del comportamiento predeterminado ofrece a la aplicación una oportunidad para mostrar un mensaje al usuario donde se le pregunta si se desea continuar.

En las siguientes secciones se describe cómo continuar con el proceso de actualización cuando se ha cancelado un paso del proceso.

Descarga e interpretación del archivo descriptor de actualización

El objeto ApplicationUpdater distribuye el evento `checkForUpdate` antes de que comience el proceso de actualización y justo antes de que updater intente descargar el archivo descriptor de actualización. Si cancela el comportamiento predeterminado del evento `checkForUpdate`, updater no descargará el archivo descriptor de actualización. Puede llamar al método `checkForUpdate()` para reanudar el proceso de actualización:

```
appUpdater.checkForUpdate();
```

Al llamar al método `checkForUpdate()`, updater interpreta y descarga de forma asincrónica el archivo descriptor de actualización. Como resultado de la llamada al método `checkForUpdate()`, el objeto updater puede distribuir los siguientes eventos:

- `StatusUpdateEvent.UPDATE_STATUS`: updater ha descargado e interpretado el archivo descriptor de actualización correctamente. Este evento cuenta con las siguientes propiedades:
  - `available`: valor booleano. Se establece en `true` si existe una versión distinta disponible a la de la aplicación actual; de lo contrario, se establece en `false` (la versión es la misma).
  - `version`: cadena. La versión del archivo descriptor de la aplicación del archivo de actualización.
  - `details`: conjunto. Si no existen versiones localizadas de la descripción, este conjunto devuelve una cadena vacía (`""`) como primer elemento y la descripción como segundo elemento.

Si existen varias versiones de la descripción (en el archivo descriptor de actualización), el conjunto contiene varios subconjuntos. Cada conjunto dispone de dos elementos: el primero es un código de idioma (como, por ejemplo, `"en"`) y el segundo es la descripción correspondiente (una cadena) para ese idioma. Consulte "Definición del archivo descriptor de actualización y adición del archivo de AIR al servidor web" en la página 115.

- `StatusUpdateErrorEvent.UPDATE_ERROR`: hubo un error y updater no pudo descargar ni interpretar el archivo descriptor de actualización.

Descarga del archivo de actualización de AIR

El objeto ApplicationUpdater distribuye el evento `updateStatus` una vez que updater descarga e interpreta correctamente el archivo descriptor de actualización. El comportamiento predeterminado es comenzar a descargar el archivo de actualización, si está disponible. Si cancela este comportamiento, puede llamar al método `downloadUpdate()` para que se reanude el proceso de actualización:

```
appUpdater.downloadUpdate();
```

Al llamar a este método, updater descarga de forma asincrónica la versión de actualización del archivo de AIR.

El método `downloadUpdate()` puede distribuir los siguientes eventos:

- `UpdateEvent.DOWNLOAD_START`: se ha establecido la conexión con el servidor. Al utilizar la biblioteca ApplicationUpdaterUI, este evento muestra un cuadro de diálogo con una barra de progreso para realizar un seguimiento del curso de la descarga.
- `ProgressEvent.PROGRESS`: se distribuye periódicamente conforme progresa la descarga del archivo.
- `DownloadErrorEvent.DOWNLOAD_ERROR`: se distribuye si se produce un error al conectar o descargar el archivo de actualización. También se distribuye para estados HTTP no válidos; por ejemplo, "404 - File not found" (No se encontró el archivo). Este evento cuenta con una propiedad `errorID`, un entero que define información de error adicional. Una propiedad adicional `subErrorID` puede incluir más información de error.
- `UpdateEvent.DOWNLOAD_COMPLETE`: updater ha descargado e interpretado correctamente el archivo descriptor de actualización. Si no cancela este evento, la versión de ApplicationUpdater procede a instalar la versión de actualización. En la versión de ApplicationUpdaterUI, se muestra un cuadro de diálogo al usuario en el que puede optar por continuar con el proceso.

Actualización de la aplicación

El objeto ApplicationUpdater distribuye el evento `downloadComplete` cuando finaliza la descarga del archivo de actualización. Si cancela el comportamiento predeterminado, puede llamar al método `installUpdate()` para que se reanude el proceso de actualización:

```
appUpdater.installUpdate(file);
```

Al llamar a este método, updater instala una versión de actualización del archivo de AIR. El método incluye un parámetro, `file`, que es un objeto File que hace referencia al archivo de AIR para usar como actualización.

El objeto ApplicationUpdater puede distribuir el evento `beforeInstall` como resultado de la llamada al método `installUpdate()`:

- `UpdateEvent.BEFORE_INSTALL`: se distribuye justo antes de la instalación de la actualización. En ocasiones, resulta útil evitar la instalación de la actualización en este momento, de modo que el usuario pueda completar el trabajo actual antes de que continúe la actualización. Con la llamada al método `preventDefault()` del objeto Event se pospone la instalación hasta el siguiente reinicio y no puede comenzar ningún proceso de actualización adicional. (Se incluyen actualizaciones que podrían aparecer con la llamada al método `checkNow()` o debido a la comprobaciones periódicas.)

Instalación desde un archivo arbitrario de AIR

Puede llamar al método `installFromAIRFile()` para instalar la versión de actualización desde un archivo de AIR en el equipo del usuario:

```
appUpdater.installFromAIRFile();
```

Este método hace que updater instale una versión de actualización de la aplicación desde el archivo de AIR.

El método `installFromAIRFile()` puede distribuir los siguientes eventos:

- `StatusFileUpdateEvent.FILE_UPDATE_STATUS`: se distribuye una vez que ApplicationUpdater ha validado correctamente el archivo enviado utilizando el método `installFromAIRFile()`. Este evento tiene las propiedades siguientes:
  - `available`: se establece en `true` si existe una versión distinta disponible a la de la aplicación actual; de lo contrario, se establece en `false` (las versiones son las mismas).
  - `version`: cadena que representa la nueva versión disponible.
  - `path`: representa la ruta nativa del archivo de actualización.

  Puede cancelar este evento si la propiedad disponible del objeto StatusFileUpdateEvent se define como `true`. La cancelación del evento implica que no continúe la actualización. Llame al método `installUpdate()` para continuar con la actualización cancelada.
- `StatusFileUpdateErrorEvent.FILE_UPDATE_ERROR`: hubo un error y updater no pudo instalar la aplicación de AIR.

Cancelación del proceso de actualización

Puede llamar al método `cancelUpdate()` para cancelar el proceso de actualización:

```
appUpdater.cancelUpdate();
```

Este método cancela todas las descargas pendientes, elimina todos los archivos descargados incompletos y reinicia el temporizador de comprobación periódica.

El método no realiza ninguna operación si el objeto updater se está inicializando.

Localización de la interfaz ApplicationUpdaterUI

La clase ApplicationUpdaterUI proporciona una interfaz de usuario predeterminada para el proceso de actualización. Esto incluye cuadros de diálogo que permiten al usuario iniciar y cancelar el proceso y llevar a cabo otras operaciones relacionadas.

El elemento `description` del archivo descriptor de actualización permite definir la descripción de la aplicación en varios idiomas. Emplee varios elementos `text` que definan atributos `lang`, tal y como se muestra a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
     <update xmlns="http://ns.adobe.com/air/framework/update/description/1.0">
       <version>1.1a1</version>
       <url>http://example.com/updates/sample_1.1a1.air</url>
       <description>
           <text xml:lang="en">English description</text>
           <text xml:lang="fr">French description</text>
           <text xml:lang="ro">Romanian description</text>
       </description>
    </update>
```

El marco de actualización utiliza la descripción que mejor se adapta a la cadena de localización del usuario final. Para obtener más información, consulte Definición del archivo descriptor de actualización y adición del archivo de AIR al servidor web.

Los desarrolladores de Flex puede añadir directamente un nuevo idioma al paquete`"ApplicationUpdaterDialogs"`.

Los desarrolladores de JavaScript pueden llamar al método `addResources()` del objeto updater. Este método agrega de forma dinámica un nuevo paquete de recursos para un idioma. El paquete de recursos define cadenas localizadas para un idioma. Estas cadenas se utilizan en distintos campos de texto de cuadro de diálogo.

Los desarrolladores de JavaScript pueden emplear la propiedad `localeChain` de la clase ApplicationUpdaterUI para definir la cadena de configuraciones regionales empleada en la interfaz de usuario. Generalmente sólo los desarrolladores de JavaScript (HTML) utilizan esta propiedad. Los desarrolladores de Flex pueden usar ResourceManager para administrar la cadena de configuraciones regionales.

Por ejemplo, el siguiente código de JavaScript define paquetes de recursos para rumano y húngaro:

```
appUpdater.addResources("ro_RO",
                {titleCheck: "Titlu", msgCheck: "Mesaj", btnCheck: "Buton"});
appUpdater.addResources("hu", {titleCheck: "Cím", msgCheck: "Üzenet"});
var languages = ["ro", "hu"];
languages = languages.concat(air.Capabilities.languages);
var sortedLanguages = air.Localizer.sortLanguagesByPreference(languages,
                    air.Capabilities.language,
                    "en-US");
sortedLanguages.push("en-US");
appUpdater.localeChain = sortedLanguages;
```

Para obtener más información, consulte la descripción del método `addResources()` de la clase ApplicationUpdaterUI en la referencia del lenguaje.

Capítulo 15: Lectura de la configuración de una aplicación

En tiempo de ejecución se pueden obtener las propiedades del archivo descriptor de la aplicación y el ID del editor de una aplicación. Estos datos se definen en las propiedades `applicationDescriptor` y `publisherID` del objeto NativeApplication.

Lectura del archivo descriptor de la aplicación

Para leer como objeto XML el archivo descriptor de la aplicación que está en ejecución, obtenga la propiedad `applicationDescriptor` del objeto NativeApplication.

Ejemplo de ActionScript 3.0:

```
var appXml:XML = NativeApplication.nativeApplication.applicationDescriptor;
```

Ejemplo de JavaScript:

```
var appXml:XML = air.ativeApplication.nativeApplication.applicationDescriptor;
```

En ActionScript 3.0, se puede acceder a los datos del descriptor de la aplicación como objeto XML (E4X), tal y como se muestra a continuación.

```
var appXml:XML = NativeApplication.nativeApplication.applicationDescriptor;
var ns:Namespace = appXml.namespace();
var appId = appXml.ns::id[0];
var appVersion = appXml.ns::version[0];
var appName = appXml.ns::filename[0];
air.trace("appId:", appId);
air.trace("version:", appVersion);
air.trace("filename:", appName);
var xmlString = air.NativeApplication.nativeApplication.applicationDescriptor;
```

En JavaScript, se puede utilizar un objeto DOMParser para analizar los datos, tal y como se muestra en el ejemplo siguiente:

```
var xmlString = air.NativeApplication.nativeApplication.applicationDescriptor;
var appXml = new DOMParser();
var xmlobject = appXml.parseFromString(xmlString, "text/xml");
var root = xmlobject.getElementsByTagName('application')[0];
var appId = root.getElementsByTagName("id")[0].firstChild.data;
var appVersion = root.getElementsByTagName("version")[0].firstChild.data;
var appName = root.getElementsByTagName("filename")[0].firstChild.data;
air.trace("appId:", appId);
air.trace("version:", appVersion);
air.trace("filename:", appName);
```

Para obtener más información, consulte “Estructura del archivo descriptor de la aplicación” en la página 71.

Obtención de los identificadores de la aplicación y del editor

Los ID de la aplicación y del editor identifican una aplicación de AIR de forma exclusiva. El ID de la aplicación se especifica en el elemento `<id>` del descriptor de la aplicación. El ID del editor se obtiene del certificado utilizado para firmar el paquete de instalación de AIR.

El ID de la aplicación puede leerse en la propiedad `id` del objeto NativeApplication, como se muestra en el siguiente código:

Ejemplo de ActionScript 3.0:

```
trace(NativeApplication.nativeApplication.applicationID);
```

Ejemplo de JavaScript:

```
air.trace(air.NativeApplication.nativeApplication.applicationID);
```

El ID del editor puede leerse en la propiedad `publisherID` del objeto NativeApplication:

Ejemplo de ActionScript 3.0:

```
trace(NativeApplication.nativeApplication.publisherID);
```

Ejemplo de ActionScript 3.0:

```
air.trace(air.NativeApplication.nativeApplication.publisherID);
```

Nota: *cuando se ejecuta una aplicación de AIR con ADL, no tendrá ID de editor a menos que se le asigne uno temporalmente utilizando el indicador* `-pubID` *en la línea de comandos de ADL.*

El ID del editor para una aplicación instalada también aparece en el archivo `META-INF/AIR/publisherid` del directorio de instalación de la aplicación.

Para obtener más información, consulte “Identificador del editor de AIR” en la página 98.

Capítulo 16: Visualización de código fuente

Del mismo modo que un usuario puede ver el código fuente de una página HTML en un navegador web, los usuarios pueden ver el código fuente de una aplicación de AIR basada en HTML. El SDK de Adobe® AIR® incluye un archivo AIRSourceViewer.js JavaScript que se puede utilizar en la aplicación para mostrar fácilmente el código fuente a los usuarios finales.

Carga, configuración y apertura del visor de código fuente

El código del visor de código fuente se incluye en un archivo JavaScript, AIRSourceViewer.js, que se incluye en el directorio frameworks del SDK de AIR. Para utilizar el visor de código fuente en su aplicación, copie AIRSourceViewer.js en el directorio project de la aplicación y cargue el archivo mediante una etiqueta de script en el archivo HTML principal en su aplicación:

```
<script type="text/javascript" src="AIRSourceViewer.js"></script>
```

El archivo AIRSourceViewer.js define una clase, SourceViewer, a la que se puede acceder desde el código JavaScript llamando a `air.SourceViewer`.

La clase SourceViewer define tres métodos: `getDefault()`, `setup()` y `viewSource()`.

Método	Descripción
`getDefault()`	Método estático. Devuelve una instancia de SourceViewer, que se puede utilizar para llamar a los otros métodos.
`setup()`	Aplica las opciones de configuración al visor de código fuente. Para obtener más información, consulte "Configuración del visor de código fuente" en la página 126
`viewSource()`	Abre una nueva ventana en la que el usuario puede examinar y abrir archivos de origen de la aplicación host.

***Nota:** el código que utiliza el visor de código fuente debe estar en el entorno limitado de seguridad de la aplicación (en un archivo del directorio de la aplicación).*

Por ejemplo, el siguiente código JavaScript crea una instancia de un objeto Source Viewer y abre la ventana de Source Viewer donde se incluyen todos los archivos de origen:

```
var viewer = air.SourceViewer.getDefault();
viewer.viewSource();
```

Configuración del visor de código fuente

El método `config()` aplica la configuración dada en el visor de código fuente. Este método adopta un parámetro: `configObject`. El objeto `configObject` dispone de propiedades que definen las opciones de configuración del visor de código fuente. Entre las propiedades se incluyen `default`, `exclude`, `initialPosition`, `modal`, `typesToRemove` y `typesToAdd`.

default
Cadena que especifica la ruta relativa al archivo inicial que se mostrará en el visor de código fuente.

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente con el archivo index.html como archivo inicial mostrado:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.default = "index.html";
viewer.viewSource(configObj);
```

exclude
Conjunto de cadenas que especifican los archivos o directorios que se excluirán de la lista del visor de código fuente. Las rutas son relativas al directorio de la aplicación. Los caracteres comodines no se admiten.

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente donde se incluyen todos los archivos de origen, excepto AIRSourceViewer.js y los archivos de los subdirectorios Images y Sounds:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.exclude = ["AIRSourceViewer.js", "Images" "Sounds"];
viewer.viewSource(configObj);
```

initialPosition
Conjunto que incluye dos números, que especifican las coordenadas iniciales x e y de la ventana del visor de código fuente.

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente con las coordenadas de la pantalla [40, 60] (X = 40, Y = 60):

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.initialPosition = [40, 60];
viewer.viewSource(configObj);
```

modal
Valor booleano que especifica si el visor de código fuente debe ser una ventana modal (true) o no modal (false). De forma predeterminada, la ventana del visor de código fuente es modal.

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente de modo que el usuario pueda interactuar tanto con la ventana del visor como con cualquier ventana de la aplicación:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.modal = false;
viewer.viewSource(configObj);
```

typesToAdd
Conjunto de cadenas que especifica los tipos de archivo que se incluirán en la lista del visor de código fuente, además de los tipos predeterminados incluidos.

De forma predeterminada, la lista Visor de origen incluye los siguientes tipos de archivo:

- Archivos de texto: TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG.
- Archivo de imagen: JPG, JPEG, PNG, GIF.

Si no se especifica ningún valor, se incluyen todos los tipos predeterminados (excepto los especificados en la propiedad `typesToExclude`).

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente con los archivos VCF y VCARD:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToAdd = ["text.vcf", "text.vcard"];
viewer.viewSource(configObj);
```

Para cada tipo de archivo que se incluya, se debe especificar "text" (para los tipos de archivo de texto) o "image" (para los tipos de archivo de imagen).

typesToExclude
Conjunto de cadenas que especifican los tipos de archivo que se excluirán del visor de código fuente.

De forma predeterminada, la lista Visor de origen incluye los siguientes tipos de archivo:

- Archivos de texto: TXT, XML, MXML, HTM, HTML, JS, AS, CSS, INI, BAT, PROPERTIES, CONFIG.
- Archivo de imagen: JPG, JPEG, PNG, GIF.

Por ejemplo, el siguiente código JavaScript abre la ventana del visor de código fuente sin incluir los archivos GIF y XML:

```
var viewer = air.SourceViewer.getDefault();
var configObj = {};
configObj.typesToExclude = ["image.gif", "text.xml"];
viewer.viewSource(configObj);
```

Para cada tipo de archivo que se incluya, se debe especificar `"text"` (para los tipos de archivo de texto) o `"image"` (para los tipos de archivo de imagen).

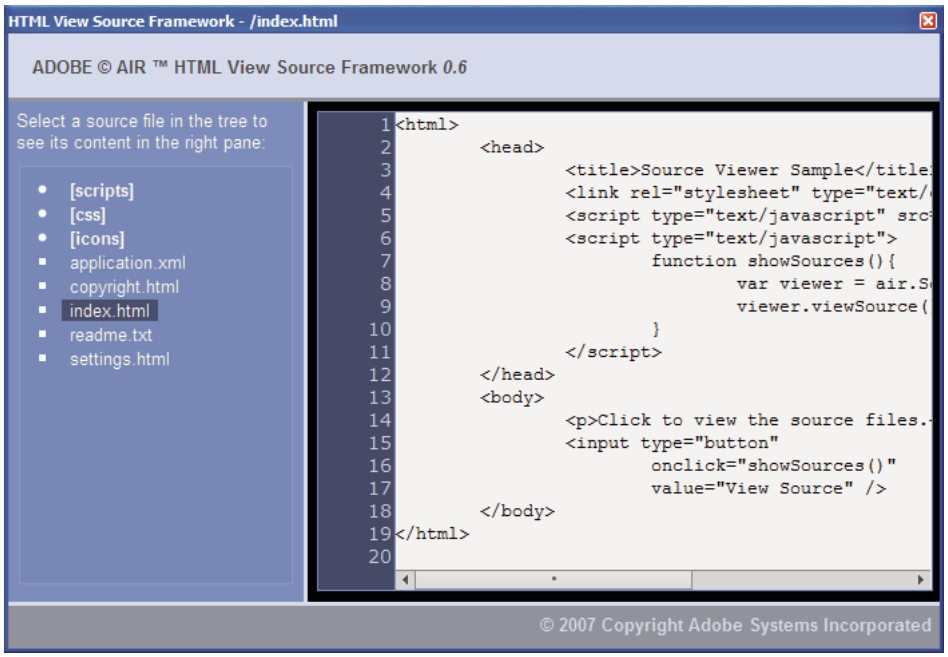
Apertura del visor de código fuente

Se debe incluir un elemento de interfaz de usuario como, por ejemplo, un vínculo, botón o comando de menú, que llame al código del visor de código fuente cuando el usuario lo seleccione. Por ejemplo, la siguiente aplicación sencilla abre el visor de código fuente cuando el usuario hace clic en un vínculo:

```
<html>
    <head>
        <title>Source Viewer Sample</title>
        <script type="text/javascript" src="AIRSourceViewer.js"></script>
        <script type="text/javascript">
            function showSources(){
                var viewer = air.SourceViewer.getDefault();
                viewer.viewSource()
            }
        </script>
    </head>
    <body>
        <p>Click to view the source files.</p>
        <input type="button"
            onclick="showSources()"
            value="View Source" />
    </body>
</html>
```

Interfaz de usuario del visor de código fuente

Si la aplicación llama al método `viewSource()` de un objeto SourceViewer, la aplicación de AIR abre una ventana del visor de código fuente. La ventana incluye una lista de directorios y archivos de origen (en la parte izquierda) y un área de visualización que muestra el código fuente del archivo seleccionado (en la parte derecha):



Los directorios se incluyen entre corchetes. El usuario puede hacer clic en una llave para expandir o contraer la lista de un directorio.

El visor de código fuente puede mostrar el código de los archivos de texto con extensiones reconocidas (por ejemplo, HTML, HTML, JS, TXT, XML, etc.) o de archivos de imagen con extensiones de imagen reconocidas (JPG, JPEG, PNG y GIF). Si el usuario selecciona un archivo que no dispone de una extensión reconocida, aparece el mensaje de error "Cannot retrieve text content from this filetype" (No se puede recuperar el contenido de texto de este tipo de archivo).

Todos los archivos de origen excluidos mediante el método `setup()` no se incluyen en la lista (consulte "Carga, configuración y apertura del visor de código fuente" en la página 126).

Capítulo 17: Depuración con el introspector HTML de AIR

EL SDK de Adobe® AIR® incluye un archivo de JavaScript AIRIntrospector.js que se puede incluir en la aplicación para ayudar a depurar aplicaciones basadas en HTML.

Introspector de AIR

El introspector de aplicaciones HTML/JavaScript de Adobe AIR (denominado AIR HTML Introspector) ofrece funciones útiles para ayudar en el desarrollo y depuración de aplicaciones basadas en HTML:

- Incluye una herramienta introspectora que permite señalar un elemento de la interfaz de usuario en la aplicación y ver su marcado y propiedades DOM.
- Incluye una consola para enviar referencias de objetos para introspección y es posible ajustar valores de propiedad y ejecutar código JavaScript. Asimismo, se pueden serializar objetos en la consola, lo que supone limitaciones para editar los datos. También se puede copiar y guardar texto desde la consola.
- Incluye una vista de árbol para las funciones y propiedades DOM.
- Permite editar atributos y nodos de texto para elementos DOM.
- Realiza listados de vínculos, estilos CSS, imágenes y archivos de JavaScript cargados en la aplicación.
- Permite visualizar el código fuente HTML inicial y el código fuente de marcado para la interfaz de usuario.
- Permite acceder a archivos en el directorio de la aplicación. (Esta función sólo está disponible en la consola AIR HTML Introspector abierta para el entorno limitado de la aplicación. No está disponible para las consolas abiertas para el contenido del entorno limitado que no pertenece a la aplicación.)
- Incluye un visor para objetos XMLHttpRequest y sus propiedades, entre las que se incluyen `responseText` y `responseXML` (si están disponibles).
- Puede realizar búsquedas por texto coincidente en los archivos y el código fuente.

Carga del código del introspector de AIR

El código del introspector de AIR se incluye en un archivo de JavaScript, AIRIntrospector.js, incluido en el directorio frameworks del SDK de AIR. Para utilizar el introspector de AIR en la aplicación, copie AIRIntrospector.js en el directorio project de la aplicación y cargue el archivo mediante una etiqueta de script en el archivo HTML principal de la aplicación:

```
<script type="text/javascript" src="AIRIntrospector.js"></script>
```

Asimismo, incluya en archivo en todos los archivos HTML que se correspondan con diferentes ventanas nativas de la aplicación.

Importante: *incluya el archivo AIRIntrospector.js únicamente al desarrollar y depurar la aplicación. Elimínelo de la aplicación de AIR empaquetada que se distribuya.*

El archivo AIRIntrospector.js define una clase, Console, a la que se puede acceder desde el código JavaScript, llamando a `air.Introspector.Console`.

Nota: *el código que utilice el introspector de AIR debe estar en el entorno limitado de seguridad de la aplicación (en un archivo del directorio de la aplicación).*

Inspección de un objeto en la ficha Console (Consola)

La clase Console define cinco métodos: `log()`, `warn()`, `info()`, `error()` y `dump()`.

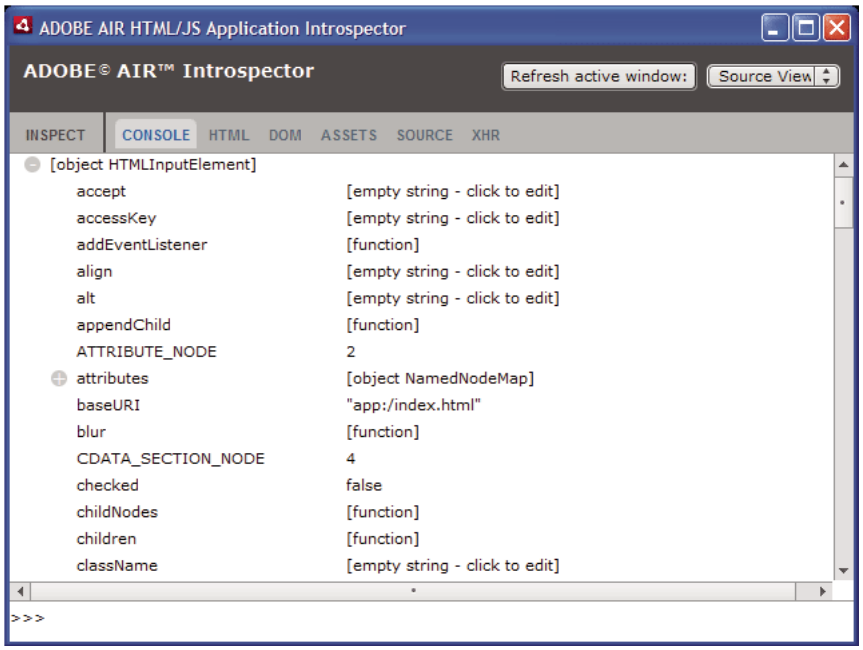
Los métodos `log()`, `warn()`, `info()`, and `error()` permiten enviar un objeto a la ficha Console (Consola). El método más básico es `log()`. El siguiente código envía un objeto simple, representado mediante la variable `test`, a la ficha Console (Consola):

```
var test = "hello";
air.Introspector.Console.log(test);
```

No obstante, resulta más útil enviar un objeto complejo a la ficha. Por ejemplo, la siguiente página HTML incluye un botón (`btn1`) que llama a una función que, a su vez, envía el propio objeto del botón a la ficha Console (Consola):

```
<html>
    <head>
        <title>Source Viewer Sample</title>
        <script type="text/javascript" src="scripts/AIRIntrospector.js"></script>
        <script type="text/javascript">
            function logBtn()
            {
                var button1 = document.getElementById("btn1");
                air.Introspector.Console.log(button1);
            }
        </script>
    </head>
    <body>
        <p>Click to view the button object in the Console.</p>
        <input type="button" id="btn1"
            onclick="logBtn()"
            value="Log" />
    </body>
</html>
```

Cuando se hace clic en el botón, la ficha Console (Consola) muestra el objeto btn1 y se puede expandir la vista de árbol del objeto para inspeccionar sus propiedades:



Se puede editar una propiedad del objeto haciendo clic en la lista situada a la derecha del nombre de la propiedad y modificando el listado de texto.

Los métodos `info()`, `error()` y `warn()` son similares al método `log()`. Sin embargo, cuando se llama a estos métodos, la consola muestra un icono al principio de la línea:

Método	**Icono**
`info()`	
`error()`	
`warn()`	

Los métodos `log()`, `warn()`, `info()` y `error()` envían una referencia sólo a un objeto real, por lo que las propiedades disponibles son las que se muestran en el momento de la visualización. Si desea serializar el objeto real, utilice el método `dump()`. El método cuenta con dos parámetros:

Parámetro	**Descripción**
`dumpObject`	Objeto que se va a serializar.
`levels`	Número máximo de niveles que se examinarán en el árbol del objeto (además del nivel de raíz). El valor predeterminado es 1 (lo que significa que se muestra un nivel superior al nivel de raíz del árbol). Este parámetro es opcional.

Al llamar al método `dump()` se serializa un objeto antes de enviarlo a la ficha Console (Consola), por lo que no se pueden editar las propiedades de los objetos. Por ejemplo, considérese el fragmento de código siguiente:

```
var testObject = new Object();
testObject.foo = "foo";
testObject.bar = 234;
air.Introspector.Console.dump(testObject);
```

Cuando se ejecuta este código, la consola muestra el objeto `testObject` y sus propiedades, pero no se pueden editar los valores de las propiedades en la consola.

Configuración del introspector de AIR

La consola se puede configurar definiendo las propiedades de la variable global `AIRIntrospectorConfig`. Por ejemplo, el siguiente código JavaScript configura el introspector de AIR para que se ajusten las columnas a 100 caracteres:

```
var AIRIntrospectorConfig = new Object();
AIRIntrospectorConfig.wrapColumns = 100;
```

Asegúrese de establecer las propiedades de la variable `AIRIntrospectorConfig` antes de cargar el archivo AIRIntrospector.js (mediante una etiqueta `script`).

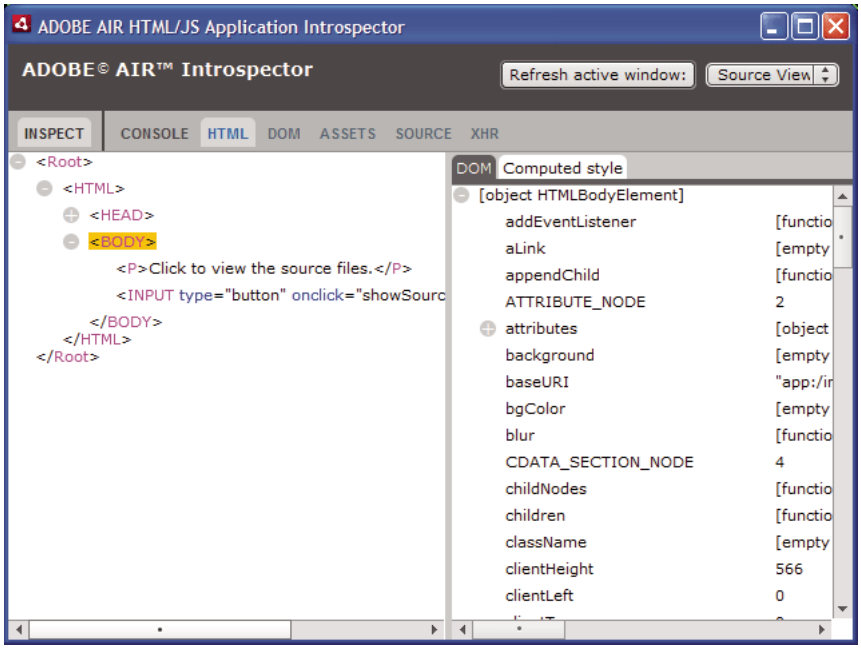
Existe ocho propiedades de la variable `AIRIntrospectorConfig`:

Propiedad	Valor predeterminado	Descripción
`closeIntrospectorOnExit`	`true`	Establece la ventana del inspector para que se cierre cuando se cierren todas las demás ventanas de la aplicación.
`debuggerKey`	123 (tecla F12)	Código de tecla del método abreviado de teclado para mostrar y ocultar la ventana del introspector de AIR.
`debugRuntimeObjects`	`true`	Establece el introspector para que amplíe los objetos del motor de ejecución definidos en JavaScript.
`flashTabLabels`	`true`	Establece las fichas Console (Consola) y XMLHttpRequest en flash, indicando el momento en que se producen cambios (por ejemplo, cuando el texto se registre en estas fichas).
`introspectorKey`	122 (tecla F11)	Código de tecla del método abreviado de teclado para abrir el panel Inspect (Inspeccionar).
`showTimestamp`	`true`	Establece la ficha Console (Consola) para que las marcas de hora aparezcan al principio de todas las líneas.
`showSender`	`true`	Establece la ficha Console (Consola) para que muestre información sobre el objeto que envía el mensaje al principio de todas las líneas.
`wrapColumns`	2000	Número de columnas al que se ajustan los archivos de origen.

Interfaz del introspector de AIR

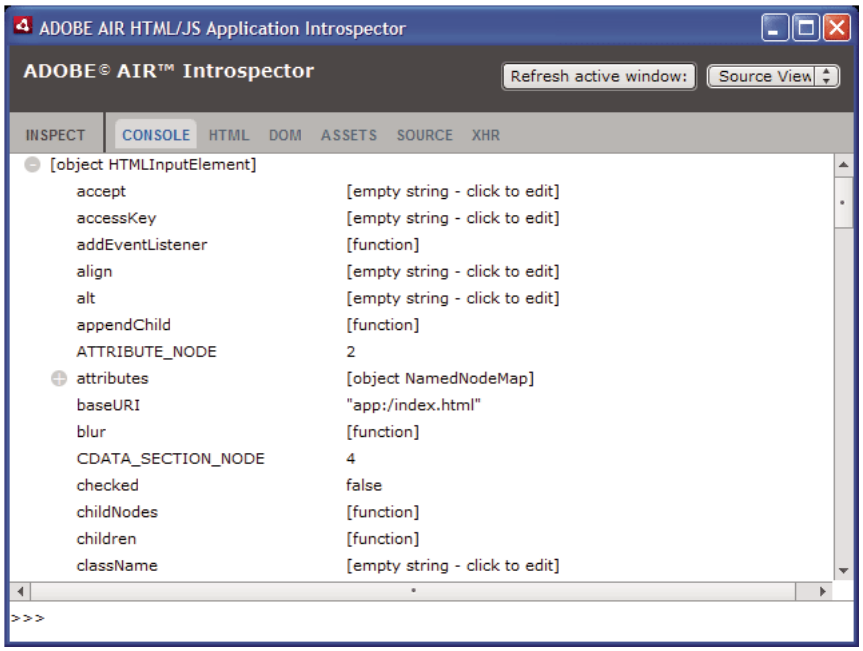
Para abrir la ventana del introspector de AIR al depurar la aplicación, presione la tecla F12 o llame a uno de los métodos de la clase Console [consulte “Inspección de un objeto en la ficha Console (Consola)” en la página 131]. Puede configurar la tecla directa para que sea una tecla distinta a F12; consulte “Configuración del introspector de AIR” en la página 133.

La ventana del introspector de AIR dispone de séis fichas: Console (Consola), HTML, DOM, Assets (Componentes), Source (Código fuente) y XHR, tal y como se muestra en la siguiente ilustración:



Ficha Console (Consola)

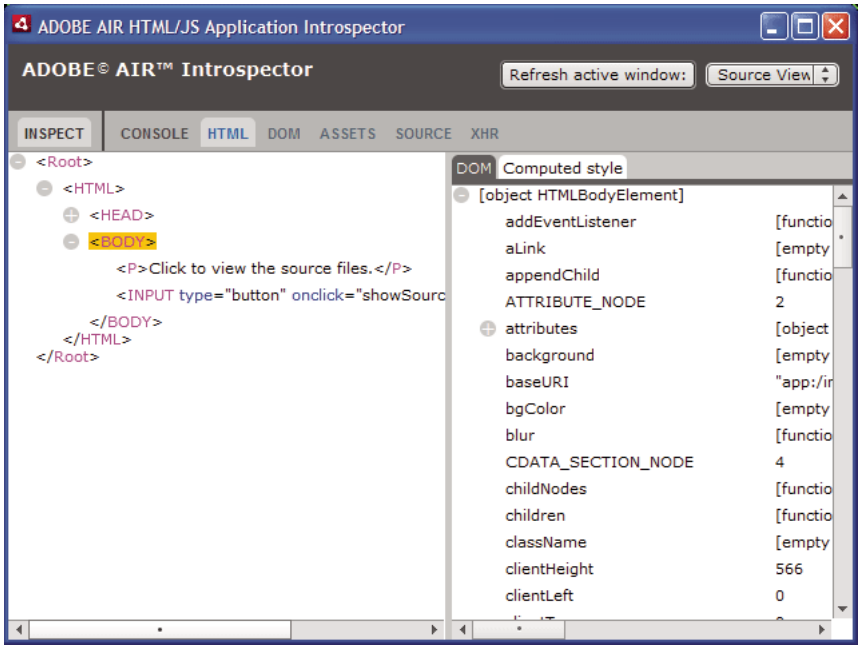
La ficha Console (Consola) muestra los valores de las propiedades transmitidas como parámetros a uno de los métodos de la clase air.Introspector.Console. Para obtener más información, consulte "Inspección de un objeto en la ficha Console (Consola)" en la página 131.



- Para borrar la consola, haga clic con el botón derecho del ratón en el texto y seleccione Clear Console (Borrar consola).
- Para guardar texto en la ficha Console (Consola) en un archivo, haga clic con el botón derecho del ratón en la ficha de consola y seleccione Save Console To File (Guardar consola en archivo).
- Para guardar texto en la ficha Console (Consola) en el portapapeles, haga clic con el botón derecho del ratón en la ficha de consola y seleccione Save Console To Clipboard (Guardar consola en portapapeles). Para copiar sólo texto seleccionado en el portapapeles, haga clic con el botón derecho del ratón en el texto y seleccione Copy (Copiar).
- Para guardar texto de la clase Console en un archivo, haga clic con el botón derecho del ratón en la ficha de consola y seleccione Save Console To File (Guardar consola en archivo).
- Para buscar texto coincidente mostrado en la ficha, presione Ctrl+F en Windows o Comando+F en Mac OS. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Ficha HTML

La ficha HTML permite ver todo el DOM de HTML en una estructura de árbol. Haga clic en un elemento para ver sus propiedades en la parte derecha de la ficha. Haga clic en los iconos + y - para expandir y contraer un nodo en el árbol.



Se puede editar cualquier atributo o elemento de texto en la ficha HTML y el valor editado se refleja en la aplicación.

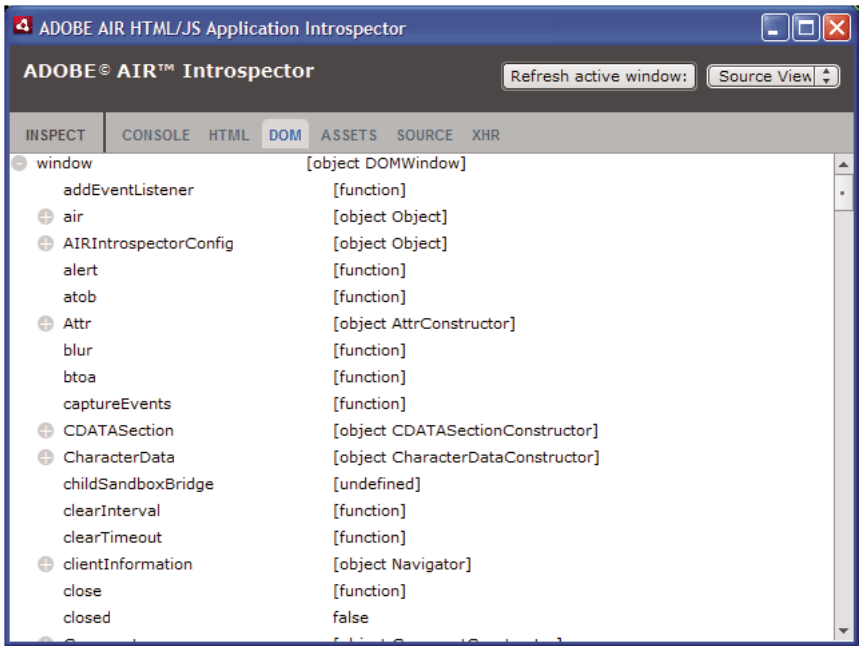
Haga clic en el botón Inspect (Inspeccionar) (situado a la izquierda de la lista de fichas de la ventana del introspector de AIR). Se puede hacer clic en cualquier elemento de la página HTML de la ventana principal y el objeto DOM asociado se muestra en la ficha HTML. Si la ventana principal está seleccionada, también se puede pulsar el método abreviado de teclado para activar y desactivar el botón Inspect (Inspeccionar). El método abreviado de teclado es F11 de forma predeterminada. Es posible configurar el método abrevado de teclado para sea una tecla distinta a F11; consulte “Configuración del introspector de AIR” en la página 133.

Haga clic en el botón Refresh Active Window (Actualizar ventana activa) (situado en la parte superior de la ventana del introspector de AIR) para actualizar los datos que se muestran en la ficha HTML.

Presione Ctrl+F en Windows o Comando+F en Mac OS para buscar texto coincidente que se muestre en la ficha. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Ficha DOM

La ficha DOM muestra los objetos de ventana en una estructura de árbol. Se puede editar cualquier propiedad string y numeric y el valor editado se refleja en la aplicación.

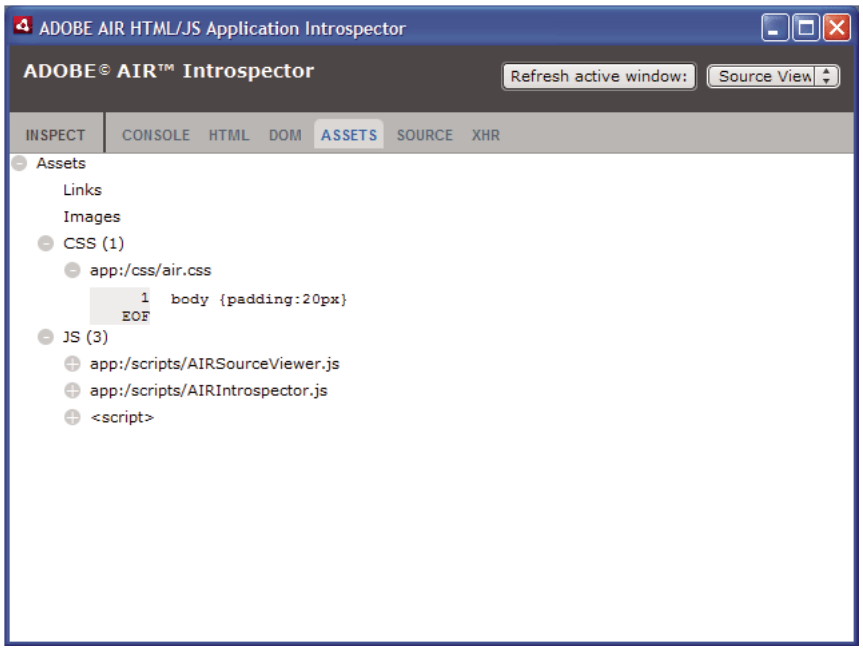


Haga clic en el botón Refresh Active Window (Actualizar ventana activa) (situado en la parte superior de la ventana del introspector de AIR) para actualizar los datos que se muestran en la ficha DOM.

Presione Ctrl+F en Windows o Comando+F en Mac OS para buscar texto coincidente que se muestre en la ficha. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Ficha Assets (Componentes)

La ficha Assets (Componentes) permite comprobar vínculos, imágenes, CSS y archivos de JavaScript cargados en la ventana nativa. Al expandir uno de estos nodos se muestra el contenido del archivo o la imagen real utilizada.



Haga clic en el botón Refresh Active Window (Actualizar ventana activa) (situado en la parte superior de la ventana del introspector de AIR) para actualizar los datos que se muestran en la ficha Assets (Componentes).

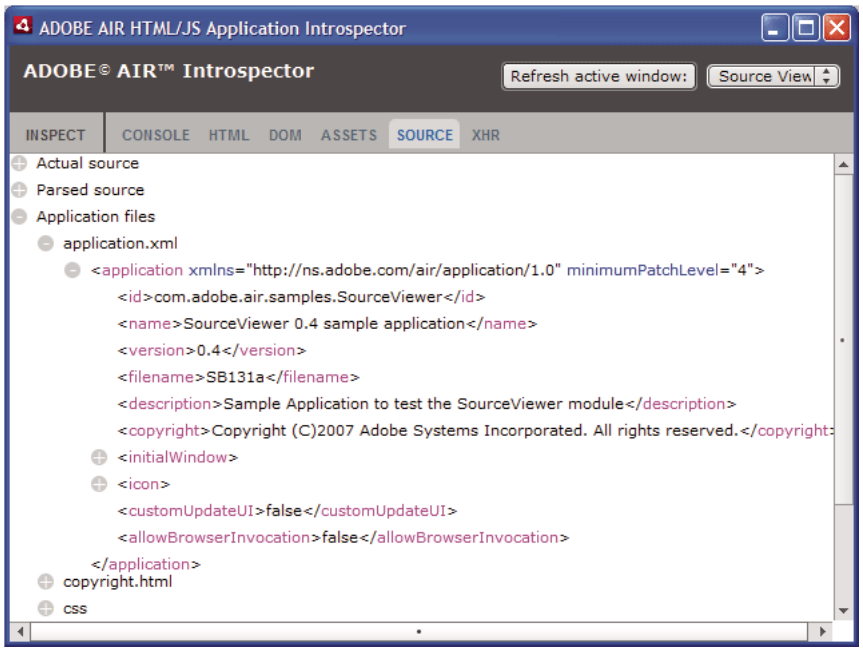
Presione Ctrl+F en Windows o Comando+F en Mac OS para buscar texto coincidente que se muestre en la ficha. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Ficha Source (Código fuente)

La ficha Source (Código fuente) incluye tres secciones:

- Código fuente real: muestra el código fuente HTML de la página cargado como contenido raíz cuando se inició la aplicación.
- Código fuente analizado: muestra el marcado actual que conforma la IU de la aplicación, que puede ser distinto al código fuente real, ya que la aplicación genera código de marcado sobre la marcha utilizando técnicas de Ajax.

- Archivos de la aplicación: incluye los archivos en el directorio de la aplicación. Esta lista sólo está disponible para el introspector de AIR cuando se inicia desde el contenido del entorno limitado de seguridad de la aplicación. En esta sección se puede ver el contenido de archivos de texto o ver imágenes.

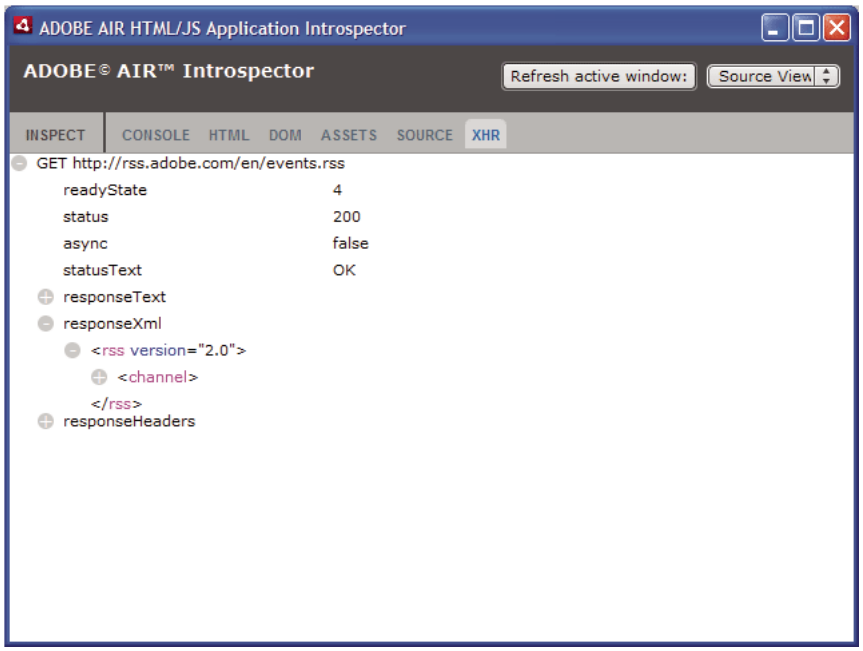


Haga clic en el botón Refresh Active Window (Actualizar ventana activa) (situado en la parte superior de la ventana del introspector de AIR) para actualizar los datos que se muestran en la ficha Source (Código fuente).

Presione Ctrl+F en Windows o Comando+F en Mac OS para buscar texto coincidente que se muestre en la ficha. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Ficha XHR

La ficha XHR intercepta toda la comunicación XMLHttpRequest en la aplicación y registra la información. Esto permite ver las propiedades XMLHttpRequest, entre las que se incluyen `responseText` y `responseXML` (cuando están disponibles) en una vista de árbol.



Presione Ctrl+F en Windows o Comando+F en Mac OS para buscar texto coincidente que se muestre en la ficha. (Las búsquedas no se realizan en los nodos de árbol que no estén visibles.)

Utilización del introspector de AIR con contenido en un entorno limitado ajeno a la aplicación

Se puede cargar contenido de un directorio de la aplicación en un iframe o frame que se asigne a un entorno limitado ajeno a la aplicación. (Consulte Seguridad HTML en Adobe AIR para desarrolladores de ActionScript o Seguridad HTML en Adobe AIR para desarrolladores de ActionScript). El introspector de AIR se puede utilizar con este contenido, pero se deben tener en cuenta las siguientes reglas:

- El archivo AIRIntrospector.js se debe incluir en el contenido del entorno limitado de la aplicación y del entorno limitado ajeno a la aplicación (iframe).
- No sobrescriba la propiedad `parentSandboxBridge`; el código del introspector de AIR usa esta propiedad. Añada propiedades según sea necesario. Por lo tanto, en lugar de escribir lo siguiente:

  ```
  parentSandboxBridge = mytrace: function(str) {runtime.trace(str)}} ;
  ```

  Utilice la siguiente sintaxis:

  ```
  parentSandboxBridge.mytrace = function(str) {runtime.trace(str)};
  ```

- Desde el contenido del entorno limitado ajeno a la aplicación no es posible abrir el introspector de AIR pulsando la tecla F12 ni llamando a uno de los métodos de la clase air.Introspector.Console. La ventana del introspector sólo se puede abrir haciendo clic en el botón Open Introspector (Abrir introspector). El botón se añade de forma predeterminada a la esquina superior derecha de iframe o frame. (Debido a las restricciones de seguridad aplicadas al contenido del entorno limitado ajeno a la aplicación, una ventana nueva sólo se puede abrir como resultado de un gesto del usuario como, por ejemplo, hacer clic en un botón.)
- Es posible abrir ventanas del introspector de AIR independientes para el entorno limitado de la aplicación y el entorno limitado ajeno a la aplicación. Los dos entornos se pueden diferenciar utilizando el título que aparece en las ventanas del introspector de AIR.
- La ficha Source (Código fuente) no muestra archivos de la aplicación cuando el introspector de AIR se ejecuta desde un entorno limitado ajeno a la aplicación.
- El introspector de AIR sólo puede consultar código en el entorno limitado desde el que se ha abierto.

Capítulo 18: Localización de aplicaciones de AIR

Adobe AIR 1.1 y posterior

Adobe® AIR® es compatible con el uso de diversos idiomas.

Para obtener información general sobre la localización de contenido en ActionScript 3.0 y la arquitectura de Flex, consulte "Localización de aplicaciones" en la guía del desarrollador de Adobe ActionScript 3.0.

Idiomas admitidos en AIR

La compatibilidad con la localización para las aplicaciones de AIR en los siguientes idiomas se introdujo en AIR 1.1:

- Chino simplificado
- Chino tradicional
- Francés
- Alemán
- Italiano
- Japonés
- Coreano
- Portugués brasileño
- Ruso
- Español

En AIR 1.5, se añadieron los siguientes idiomas:

- Checo
- Neerlandés
- Polaco
- Sueco
- Turco

Más temas de ayuda

Building multilingual Flex applications on Adobe AIR (Creación de aplicaciones de Flex multilingües en Adobe AIR; en inglés)

Building a multilingual HTML-based application (Creación de una aplicación multilingüe basada en HTML; en inglés)

Localización del nombre y la descripción en el instalador de aplicaciones de AIR

Adobe AIR 1.1 y posterior

Se pueden especificar varios idiomas para los elementos `name` y `description` en el archivo descriptor de la aplicación. En el ejemplo siguiente se especifica el nombre de la aplicación en tres idiomas (inglés, francés y alemán):

```
<name>
    <text xml:lang="en">Sample 1.0</text>
    <text xml:lang="fr">Échantillon 1.0</text>
    <text xml:lang="de">Stichprobe 1.0</text>
</name>
```

El atributo `xml:lang` de cada elemento de texto especifica un código de idioma, tal y como se define en RFC4646 (http://www.ietf.org/rfc/rfc4646.txt).

El elemento "name" sólo define el nombre de la aplicación que presenta el instalador de aplicaciones de AIR. El instalador de aplicaciones de AIR utiliza el valor localizado que mejor se corresponde con el idioma de la interfaz de usuario definido en la configuración del sistema operativo.

Se pueden especificar varias versiones de idiomas del elemento `description` en el archivo descriptor de la aplicación. Este elemento define el texto descriptivo que presenta el instalador de aplicaciones de AIR.

Estas opciones sólo se aplican a los idiomas que se ofrecen en el instalador de aplicaciones de AIR. No definen las configuraciones regionales disponibles para la aplicación instalada y en funcionamiento. Las aplicaciones de AIR pueden ofrecer interfaces de usuario compatibles con varios idiomas, incluidos los del instalador de aplicaciones de AIR y otros adicionales.

Para obtener más información, consulte “Definición de propiedades en el archivo descriptor de la aplicación” en la página 72.

Más temas de ayuda

Building multilingual Flex applications on Adobe AIR (Creación de aplicaciones de Flex multilingües en Adobe AIR; en inglés)

Building a multilingual HTML-based application (Creación de una aplicación multilingüe basada en HTML; en inglés)

Localización de contenido HTML con la arquitectura de localización de HTML de AIR

Adobe AIR 1.1 y posterior

El SDK de AIR 1.1 incluye una arquitectura de localización de HTML. El archivo JavaScript AIRLocalizer.js define la arquitectura. El archivo AIRLocalizer.js se encuentra en el directorio frameworks del SDK de AIR. Este archivo incluye la clase air.Localizer, que ofrece funciones de utilidad para la creación de aplicaciones compatibles con varias versiones localizadas.

Carga del código de la arquitectura de localización de HTML de AIR

Para utilizar la arquitectura de localización, copie el archivo AIRLocalizer.js en su proyecto. Inclúyalo en el archivo HTML principal de la aplicación con una etiqueta de script:

```
<script src="AIRLocalizer.js" type="text/javascript" charset="utf-8"></script>
```

El código JavaScript que le sigue llamar al objeto `air.Localizer.localizer`:

```
<script>
    var localizer = air.Localizer.localizer;
</script>
```

El objeto `air.Localizer.localizer` es un objeto de instancia única que define métodos y propiedades para utilizar y gestionar los recursos localizados. La clase Localizer incluye los métodos siguientes:

Método	Descripción
`getFile()`	Obtiene el texto de un paquete de recursos especificado para una configuración regional especificada. Consulte "Obtención de recursos para una configuración regional específica" en la página 150.
`getLocaleChain()`	Devuelve los idiomas de la cadena de configuraciones regionales. Consulte "Definición de la cadena de configuraciones regionales" en la página 149.
`getResourceBundle()`	Devuelve las claves del paquete y los valores correspondientes como un objeto. Consulte "Obtención de recursos para una configuración regional específica" en la página 150.
`getString()`	Obtiene la cadena de caracteres definida para un recurso. Consulte "Obtención de recursos para una configuración regional específica" en la página 150.
`setBundlesDirectory()`	Configura la ubicación del directorio de paquetes (bundles). Consulte "Personalización de las opciones de AIR HTML Localizer" en la página 148.
`setLocalAttributePrefix()`	Define el prefijo para los atributos de localización que se utilizan en los elementos DOM de HTML. Consulte "Personalización de las opciones de AIR HTML Localizer" en la página 148.
`setLocaleChain()`	Define el orden de los idiomas en la cadena de configuraciones regionales. Consulte "Definición de la cadena de configuraciones regionales" en la página 149.
`sortLanguagesByPreference()`	Ordena las configuraciones regionales en la cadena de configuraciones regionales en función del orden en que se encuentran en la configuración del sistema operativo. Consulte "Definición de la cadena de configuraciones regionales" en la página 149.
`update()`	Actualiza el DOM de HTML (o un elemento DOM) con cadenas de caracteres localizadas procedentes de la actual cadena de configuraciones regionales. Para ver una discusión sobre las cadenas de configuraciones regionales, consulte "Gestión de cadenas de configuraciones regionales" en la página 146. Para obtener más información sobre el método `update()`, consulte "Actualización de elementos DOM para utilizar la configuración regional actual" en la página 147.

La clase Localizer incluye las siguientes propiedades estáticas:

Propiedad	Descripción
`localizer`	Devuelve una referencia al objeto Localizer de instancia única para la aplicación.
`ultimateFallbackLocale`	La configuración regional que se utiliza cuando la aplicación no admite ninguna de las preferencias del usuario. Consulte "Definición de la cadena de configuraciones regionales" en la página 149.

Definición de paquetes de recursos

La arquitectura de localización de HTML lee las versiones localizadas de cadenas de caracteres en los archivos de *localización*. Un archivo de localización es una colección de valores basados en clave y serializados en un archivo de texto. A veces se le conoce por la palabra *paquete*.

Cree un subdirectorio del directorio del proyecto de la aplicación llamado "configregional". (Puede usar otro nombre; consulte "Personalización de las opciones de AIR HTML Localizer" en la página 148.) Este directorio incluirá los archivos de localización. Este directorio se conoce como el *directorio de paquetes*.

Para cada configuración regional que admita la aplicación, cree un subdirectorio del directorio de paquetes. Dé a cada subdirectorio un nombre que corresponda al código de la configuración regional. Por ejemplo: llame al directorio francés "fr" y al directorio inglés "en". Para definir una configuración regional con códigos de idioma y de país se puede utilizar guión bajo (_). Por ejemplo, el directorio de inglés estadounidense se llamaría "en_us". (También se puede utilizar un guión normal en lugar de un guión bajo: "en-us". La arquitectura de localización de HTML admite ambas formas).

No hay límite de la cantidad de archivos de recursos que se pueden añadir a un subdirectorio de configuraciones regionales. Se suele crear un archivo de localización para cada idioma (y colocar el archivo en el directorio de ese idioma). La arquitectura de localización de HTML incluye un método `getFile()` que permite leer el contenido de un archivo. Consulte "Obtención de recursos para una configuración regional específica" en la página 150.

Los archivos que tienen la extensión de archivo .properties se denominan "archivos de propiedades de localización". Se pueden utilizar para definir pares clave-valor para una configuración regional. Un archivo de propiedades define un valor de cadena en cada línea. En el siguiente ejemplo se define el valor de cadena `"Hello in English."` para una clave denominada `greeting`:

```
greeting=Hello in English.
```

Un archivo de propiedades que contiene el texto siguiente define seis pares clave-valor:

```
title=Sample Application
greeting=Hello in English.
exitMessage=Thank you for using the application.
color1=Red
color2=Green
color3=Blue
```

Este ejemplo muestra una versión en inglés del archivo de propiedades, que se guarda en el directorio en.

La versión en francés de este archivo de propiedades se coloca en el directorio fr:

```
title=Application Example
greeting=Bonjour en français.
exitMessage=Merci d'avoir utilisé cette application.
color1=Rouge
color2=Vert
color3=Bleu
```

Se pueden definir varios archivos de recursos para distintos tipos de información. Por ejemplo: un archivo legal.properties podría contener un texto jurídico estándar (como un aviso de copyright). Quizá se desee utilizar ese recurso en varias aplicaciones. Asimismo, se pueden definir archivos separados que definen el contenido localizado para distintas partes de la interfaz de usuario.

Utilice para estos archivos la codificación UTF-8 para mayor compatibilidad con distintos idiomas.

Gestión de cadenas de configuraciones regionales

Cuando la aplicación carga el archivo AIRLocalizer.js, la misma examina las configuraciones regionales que tiene definidas. Estas configuraciones regionales corresponden a los subdirectorios del directorio de paquetes (consulte "Definición de paquetes de recursos" en la página 145). Esta lista de configuraciones regionales disponibles se denomina la *cadena de configuraciones regionales*. El archivo AIRLocalizer.js ordena automáticamente la cadena de configuraciones regionales en función del orden definido en la configuración del sistema operativo. (La propiedad `Capabilities.languages` enumera los idiomas de la interfaz de usuario del sistema operativo en orden de preferencia).

De este modo, si una aplicación define recursos para las configuraciones regionales "es, "es_ES" y "es_UY", la arquitectura AIR HTML Localizer ordena en consecuencia la cadena de configuraciones regionales. Cuando se inicia una aplicación en un sistema que da aviso de "es" como configuración regional primaria, la cadena de configuraciones regionales se ordena en la secuencia `["es", "es_ES", "es_UY"]`. En este caso la aplicación busca recursos primero en el paquete "es" y después en el paquete "es_ES".

Sin embargo, si el sistema da aviso de "es_ES" como configuración regional primaria, la clasificación es `["es_ES", "es", "es_UY"]`. En este caso la aplicación busca recursos primero en el paquete "es_ES" y después en el paquete "es".

La aplicación define automáticamente la primera configuración regional de la cadena como la configuración regional predeterminada que se va a utilizar. Puede pedir al usuario que seleccione una configuración regional la primera vez que ejecuta la aplicación. Puede optar por guardar la selección en un archivo de preferencias y en adelante utilizar esa configuración regional cada vez que se inicie la aplicación.

La aplicación puede utilizar cadenas de caracteres de recurso en cualquier configuración regional de la cadena de configuraciones regionales. Si una configuración regional específica no define una cadena de caracteres de recurso, la aplicación utiliza la siguiente cadena de caracteres de recurso que coincida para otras configuraciones regionales definidas en la cadena de dichas configuraciones.

Se puede personalizar la cadena de configuraciones regionales llamando al método `setLocaleChain()` del objeto Localizer. Consulte "Definición de la cadena de configuraciones regionales" en la página 149.

Actualización de los elementos DOM con contenido localizado

Un elemento de la aplicación puede hacer referencia a un valor de clave de un archivo de propiedades de localización. En el ejemplo siguiente, el elemento `title` especifica un atributo `local_innerHTML`. La arquitectura de localización utiliza este atributo para buscar un valor localizado. De forma predeterminada, la arquitectura busca nombres de atributo que empiezan con `"local_"`. La arquitectura actualiza los atributos cuyos nombres coinciden con el texto que sigue a `"local_"`. En este caso, la arquitectura define el atributo `innerHTML` del elemento `title`. El atributo `innerHTML` utiliza el valor definido para la clave `mainWindowTitle` en el archivo de propiedades predeterminadas (default.properties):

```
<title local_innerHTML="default.mainWindowTitle"/>
```

Si la configuración regional actual no define ningún valor que coincida, la arquitectura de localización busca en el resto de la cadena de configuraciones regionales. Utiliza la siguiente configuración regional de la cadena que tenga definido un valor.

En el siguiente ejemplo el texto (el atributo `innerHTML`) del elemento `p` utiliza el valor de la clave `greeting` definido en el archivo de propiedades predeterminadas:

```
<p local_innerHTML="default.greeting" />
```

En el siguiente ejemplo el atributo del valor (y el texto mostrado) del elemento `input` utiliza el valor de la clave `btnBlue` definido en el archivo de propiedades predeterminadas:

```
<input type="button" local_value="default.btnBlue" />
```

Para actualizar el DOM de HTML para que utilice las cadenas de caracteres definidas en la cadena de configuraciones regionales actual, llame al método `update()` del objeto Localizer. Al llamar al método `update()` el objeto Localizer analiza el DOM y aplica manipulaciones donde encuentre atributos de localización (`"local_..."`):

```
air.Localizer.localizer.update();
```

Se pueden definir valores tanto para un atributo ("innerHTML", por ejemplo) como para su correspondiente atributo de localización ("local_innerHTML", por ejemplo). En este caso, la arquitectura de localización sólo sobrescribe el valor del atributo si encuentra un valor coincidente en la cadena de localización. Por ejemplo, el siguiente elemento define ambos atributos, `value` y `local_value`:

```
<input type="text" value="Blue" local_value="default.btnBlue"/>
```

También puede actualizarse un solo elemento DOM en particular. Consulte el apartado siguiente, "Actualización de elementos DOM para utilizar la configuración regional actual" en la página 147.

De forma predeterminada, AIR HTML Localizer utiliza `"local_"` como prefijo para los atributos que definen las opciones de localización para un elemento. Por ejemplo: de forma predeterminada, un atributo `local_innerHTML` define el nombre del paquete y recurso que se utiliza para el valor `innerHTML` de un elemento. También de forma predeterminada, un atributo `local_value` define el nombre del paquete y recurso que se utiliza para el atributo `value` de un elemento. Se puede configurar AIR HTML Localizer para que utilice otro prefijo de atributo en lugar de `"local_"`. Consulte "Personalización de las opciones de AIR HTML Localizer" en la página 148.

Actualización de elementos DOM para utilizar la configuración regional actual

Cuando el objeto Localizer actualiza del DOM de HTML, hace que los elementos marcados utilicen valores de atributo basados en las cadenas de caracteres definidas en la actual cadena de configuraciones regionales. Para que el localizador de HTML actualice el DOM de HTML, llame al método `update()` del objeto `Localizer`:

```
air.Localizer.localizer.update();
```

Para actualizar un solo elemento DOM especificado, páselo como parámetro al método `update()`. El método `update()` tiene un solo parámetro, `parentNode`, que es optativo. Cuando está especificado, el parámetro `parentNode` define el elemento DOM que se debe localizar. Llamar al método `update()` y especificar un parámetro `parentNode` define valores localizados para todos los elementos secundarios que especifican atributos de localización.

Por ejemplo, tomemos el siguiente elemento `div`:

```
<div id="colorsDiv">
    <h1 local_innerHTML="default.lblColors" ></h1>
    <p><input type="button" local_value="default.btnBlue" /></p>
    <p><input type="button" local_value="default.btnRed" /></p>
    <p><input type="button" local_value="default.btnGreen" /></p>
</div>
```

Para actualizar este elemento de modo que utilice cadenas de caracteres localizadas en la cadena de configuraciones regionales actual, use el código JavaScript siguiente:

```
var divElement = window.document.getElementById("colorsDiv");
air.Localizer.localizer.update(divElement);
```

Si no se encuentra un valor de clave en la cadena de configuraciones regionales, la arquitectura de localización define como valor de atributo el valor del atributo `"local_"`. Por ejemplo: supongamos que en el ejemplo anterior la arquitectura de localización no encuentra ningún valor para la clave `lblColors` (en ninguno de los archivos default.properties de la cadena de configuraciones regionales). En este caso, utiliza `"default.lblColors"` como el valor de `innerHTML`. El uso de este valor indica (al desarrollador) que faltan recursos.

El método `update()` distribuye un evento `resourceNotFound` cuando no encuentra un recurso en la cadena de configuraciones regionales. La constante `air.Localizer.RESOURCE_NOT_FOUND` define la cadena `"resourceNotFound"`. El evento tiene tres propiedades: `bundleName`, `resourceName` y `locale`. La propiedad `bundleName` es el nombre del paquete en el que no se encuentra el recurso. La propiedad `resourceName` es el nombre del recurso no disponible. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso.

El método `update()` distribuye un evento `bundleNotFound` cuando no encuentra el paquete especificado. La constante `air.Localizer.BUNDLE_NOT_FOUND` define la cadena `"bundleNotFound"`. El evento tiene tres propiedades: `bundleName` y `locale`. La propiedad `bundleName` es el nombre del paquete en el que no se encuentra el recurso. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso.

El método `update()` funciona de modo asíncrono (y distribuye los eventos `resourceNotFound` y `bundleNotFound` de forma asíncrona). El siguiente código define detectores para los eventos `resourceNotFound` y `bundleNotFound`:

```
air.Localizer.localizer.addEventListener(air.Localizer.RESOURCE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.addEventListener(air.Localizer.BUNDLE_NOT_FOUND, rnfHandler);
air.Localizer.localizer.update();
function rnfHandler(event)
{
    alert(event.bundleName + ": " + event.resourceName + ":." + event.locale);
}
function bnfHandler(event)
{
    alert(event.bundleName + ":." + event.locale);
}
```

Personalización de las opciones de AIR HTML Localizer

El método `setBundlesDirectory()` del objeto Localizer permite personalizar la ruta del directorio de paquetes. El método `setLocalAttributePrefix()` del objeto Localizer permite personalizar la ruta del directorio de paquetes y el valor de atributo que utiliza el Localizer.

El directorio de paquetes predeterminado se define como el subdirectorio de configuraciones regionales del directorio de la aplicación. Para especificar otro directorio, llame al método `setBundlesDirectory()` del objeto Localizer. Este método utiliza un parámetro, `path`, que es la ruta al directorio de paquetes deseado, en forma de cadena de caracteres. El parámetro `path` puede tener cualquiera de los valores siguientes:

- Una cadena que define una ruta relativa al directorio de la aplicación, como `"configregionales"`
- Una cadena que define una URL válida que utiliza los esquemas de URL `app`, `app-storage` o `file`, por ejemplo`"app://languages"` (*no* utilice el esquema de URL `http`)
- Un objeto File

Para obtener información sobre direcciones URL y rutas de directorio, consulte:

- Rutas a objetos File (para desarrolladores de ActionScript)
- Rutas a objetos File (para desarrolladores de HTML)

En el siguiente ejemplo, el código define como directorio de paquetes un subdirectorio "languages" del directorio de almacenamiento de la aplicación (y no el directorio de la aplicación):

```
air.Localizer.localizer.setBundlesDirectory("languages");
```

Pase una ruta válida como parámetro `path`. De lo contrario, el método emite una excepción BundlePathNotFoundError. Este error tiene a `"BundlePathNotFoundError"` como su propiedad `name` y su propiedad `message` especifica la ruta no válida.

De forma predeterminada, AIR HTML Localizer utiliza `"local_"` como prefijo para los atributos que definen las opciones de localización para un elemento. Por ejemplo, el atributo `local_innerHTML` define el nombre del paquete y recurso que se utiliza para el valor `innerHTML` del siguiente elemento `input`:

```
<p local_innerHTML="default.greeting" />
```

El método `setLocalAttributePrefix()` del objeto Localizer permite utilizar otro prefijo de atributo en lugar de `"local_"`. Este método estático utiliza un parámetro, que es la cadena de caracteres que se desea utilizar como prefijo de atributo. En el siguiente ejemplo, el código define la arquitectura de localización para que utilice "loc_" como prefijo de atributo:

```
air.Localizer.localizer.setLocalAttributePrefix("loc_");
```

Se puede personalizar el prefijo de atributo que utiliza la arquitectura de localización. Puede ser conveniente personalizar el prefijo si el valor predeterminado (`"local_"`) está en conflicto con el nombre de otro atributo que se utilice en el código. Cuando llame a este método, asegúrese de utilizar caracteres válidos para los atributos de HTML. (Por ejemplo, el valor no puede contener un carácter de espacio en blanco).

Para obtener más información sobre el uso de atributos de localización en elementos HTML, consulte "Actualización de los elementos DOM con contenido localizado" en la página 146.

Las opciones de directorio de paquetes y prefijo de atributo no persisten entre distintas sesiones de la aplicación. Si utiliza opciones personalizadas para el directorio de paquetes o el prefijo de atributo, asegúrese de configurarlas cada vez que inicie la aplicación.

Definición de la cadena de configuraciones regionales

Cuando se carga el código de AIRLocalizer.js, define automáticamente la cadena de configuraciones regionales predeterminada. Las configuraciones regionales disponibles en el directorio de paquetes y la configuración de idiomas del sistema operativo definen esta cadena de configuraciones regionales. (Para obtener más información, consulte "Gestión de cadenas de configuraciones regionales" en la página 146).

Se puede modificar la cadena de configuraciones regionales llamando al método estático `setLocaleChain()` del objeto Localizer. Por ejemplo, puede ser conveniente llamar a este método si el usuario indica una preferencia para un idioma concreto. El método `setLocaleChain()` utiliza un solo parámetro, `chain`, que es un conjunto de configuraciones regionales, por ejemplo `["fr_FR","fr","fr_CA"]`. El orden de las configuraciones locales en el conjunto define el orden en que la arquitectura busca recursos (en operaciones posteriores). Si no se encuentra un recurso para la primera configuración regional de la cadena, sigue buscando en los recursos de la otra configuración regional. Si falta el argumento `chain`, o si no es un conjunto o es un conjunto vacío, la función falla y emite una excepción IllegalArgumentsError.

El método estático `getLocaleChain()` del objeto Localizer devuelve un conjunto que enumera las configuraciones regionales de la cadena de configuraciones regionales actual.

El siguiente código lee la cadena de configuraciones regionales actual y añade dos configuraciones regionales francesas a la cabeza de la cadena:

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
```

El método `setLocaleChain()` distribuye un evento `"change"` cuando actualiza la cadena de configuraciones regionales. La constante `air.Localizer.LOCALE_CHANGE` define la cadena `"change"`. El evento tiene una propiedad, `localeChain`, que es un conjunto de códigos de configuración regional en la nueva cadena de configuraciones regionales. El siguiente código define un detector para este evento:

```
var currentChain = air.Localizer.localizer.getLocaleChain();
newLocales = ["fr_FR", "fr"];
localizer.addEventListener(air.Localizer.LOCALE_CHANGE, changeHandler);
air.Localizer.localizer.setLocaleChain(newLocales.concat(currentChain));
function changeHandler(event)
{
    alert(event.localeChain);
}
```

La propiedad estática `air.Localizer.ultimateFallbackLocale` representa la configuración regional que se utiliza cuando la aplicación no admite ninguna de las preferencias del usuario. El valor predeterminado es `"en"`. Se lo puede cambiar a otra configuración regional, como en el código siguiente:

```
air.Localizer.ultimateFallbackLocale = "fr";
```

Obtención de recursos para una configuración regional específica

El método `getString()` del objeto Localizer devuelve la cadena definida para un recurso en una configuración regional específica. No hace falta especificar un valor `locale` cuando se llama al método. En este caso el método busca en toda la cadena de configuraciones regionales y devuelve la cadena de caracteres de la primera configuración regional que proporciona el nombre del recurso especificado. El método utiliza los siguientes parámetros:

Parámetro	Descripción
`bundleName`	El paquete que contiene el recurso. Es el nombre del archivo de propiedades sin la extensión .properties. Por ejemplo: si este parámetro está definido en `"alerts"`, el código del Localizer busca en archivos de localización que tengan el nombre alerts.properties.
`resourceName`	El nombre del recurso.
`templateArgs`	Opcional. Un conjunto de cadenas para sustituir las etiquetas numeradas en la cadena de sustitución. Tomemos como ejemplo una llamada a la función en que el parámetro `templateArgs` es `["Raúl", "4"]` y la cadena del recurso coincidente es `"Hello, {0}. You have {1} new messages."`. En este caso, la función devuelve `"Hello, Raúl. You have 4 new messages."`. Para pasar por alto esta opción, pase un valor `null`.
`locale`	Opcional. El código de la configuración regional (por ejemplo: `"en"`, `"en_us"` o `"fr"`) que se debe utilizar. Si se facilita una configuración regional y no se encuentra ningún valor coincidente, el método no seguirá buscando valores en otras configuraciones regionales de la cadena. Si no se especifica ningún código de configuración regional, la función devuelve la cadena de caracteres que está en la primera configuración regional que proporciona un valor para el nombre del recurso especificado.

La arquitectura de localización puede actualizar los atributos marcados del DOM de HTML. Hay también otras formas de utilizar cadenas localizadas. Por ejemplo, se puede utilizar una cadena de caracteres en HTML generado de forma dinámica o como valor de parámetro en una llamada a una función. En el siguiente ejemplo, el código llama a la función `alert()` con la cadena de caracteres definida en el recurso `error114` del archivo de propiedades predeterminadas de la configuración regional fr_FR:

```
alert(air.Localizer.localizer.getString("default", "error114", null, "fr_FR"));
```

El método `getString()` distribuye un evento `resourceNotFound` cuando no encuentra el recurso en el paquete especificado. La constante `air.Localizer.RESOURCE_NOT_FOUND` define la cadena `"resourceNotFound"`. El evento tiene tres propiedades: `bundleName`, `resourceName` y `locale`. La propiedad `bundleName` es el nombre del paquete en el que no se encuentra el recurso. La propiedad `resourceName` es el nombre del recurso no disponible. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso.

El método `getString()` distribuye un evento `bundleNotFound` cuando no encuentra el paquete especificado. La constante `air.Localizer.BUNDLE_NOT_FOUND` define la cadena `"bundleNotFound"`. El evento tiene tres propiedades: `bundleName` y `locale`. La propiedad `bundleName` es el nombre del paquete en el que no se encuentra el recurso. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso.

El método `getString()` funciona de modo asíncrono (y distribuye los eventos `resourceNotFound` y `bundleNotFound` de forma asíncrona). El siguiente código define detectores para los eventos `resourceNotFound` y `bundleNotFound`:

```
air.Localizerlocalizer.addEventListener(air.Localizer.RESOURCE_NOT_FOUND, rnfHandler);
air.Localizerlocalizer.addEventListener(air.Localizer.BUNDLE_NOT_FOUND, bnfHandler);
var str = air.Localizer.localizer.getString("default", "error114", null, "fr_FR");
function rnfHandler(event)
{
    alert(event.bundleName + ": " + event.resourceName + ":." + event.locale);
}
function bnfHandler(event)
{
    alert(event.bundleName + ":." + event.locale);
}
```

El método `getResourceBundle()` del objeto Localizer devuelve un paquete especificado para una configuración regional determinada. El valor devuelto del método es un objeto con propiedades que coinciden con las claves del paquete. (Si la aplicación no puede encontrar el paquete especificado, el método devuelve `null`.)

El método adopta dos parámetros: `locale` y `bundleName`.

Parámetro	Descripción
`locale`	Configuración reg¡onal (p. ej. `"fr"`).
`bundleName`	Nombre del paquete.

Por ejemplo, el siguiente código llama al método `document.write()` para cargar el paquete predeterminado para la configuración regional fr. A continuación llama al método `document.write()` para escribir valores de las claves `str1` y `str2` en ese paquete:

```
var aboutWin = window.open();
var bundle = localizer.getResourceBundle("fr", "default");
aboutWin.document.write(bundle.str1);
aboutWin.document.write("<br/>");
aboutWin.document.write(bundle.str2);
aboutWin.document.write("<br/>");
```

El método `getResourceBundle()` distribuye un evento `bundleNotFound` cuando no encuentra el paquete especificado. La constante `air.Localizer.BUNDLE_NOT_FOUND` define la cadena `"bundleNotFound"`. El evento tiene tres propiedades: `bundleName` y `locale`. La propiedad `bundleName` es el nombre del paquete en el que no se encuentra el recurso. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso.

El método `getFile()` del objeto Localizer devuelve el contenido de un paquete, en forma de cadena, para una configuración regional determinada. El archivo del paquete se lee como archivo UTF-8. El método incluye los siguientes parámetros:

Parámetro	Descripción
`resourceFileName`	Nombre del archivo del recurso (por ejemplo, `"about.html"`).

`templateArgs`	Opcional. Un conjunto de cadenas para sustituir las etiquetas numeradas en la cadena de sustitución. Tomemos como ejemplo una llamada a la función en que el parámetro `templateArgs` es `["Raúl", "4"]` y el archivo del recurso coincidente contiene dos líneas:
	`<html>`
	`<body>Hello, {0}. You have {1} new messages.</body>`
	`</html>`
	En este caso, la función devuelve una cadena de dos líneas:
	`<html>`
	`<body>Hello, Raúl. You have 4 new messages. </body>`
	`</html>`

`locale`	Código de la configuración regional (por ejemplo: `"es_ES"`) que se va a utilizar. Si se facilita una configuración regional y no se encuentra ningún archivo coincidente, el método no seguirá buscando en otras configuraciones regionales de la cadena. Si no se especifica *ningún* código de configuración regional, la función devuelve el texto de la primera configuración regional de la cadena que tenga un archivo que coincida con el nombre de archivo del recurso, `resourceFileName`.

En el siguiente ejemplo, el código llama al método `document.write()` utilizando el contenido del archivo about.html file de la configuración regional fr:

```
var aboutWin = window.open();
var aboutHtml = localizer.getFile("about.html", null, "fr");
aboutWin.document.close();
aboutWin.document.write(aboutHtml);
```

El método `getFile()` distribuye un evento `fileNotFound` cuando no encuentra un recurso en la cadena de configuraciones regionales. La constante `air.Localizer.FILE_NOT_FOUND` define la cadena `"resourceNotFound"`. El método `getFile()` funciona de modo asíncrono (y distribuye el evento `fileNotFound` de forma asíncrona). El evento tiene tres propiedades: `bundleName` y `locale`. La propiedad `fileName` es el nombre del archivo que no se encuentra. La propiedad `locale` es el nombre de la configuración regional en la que no se encuentra el recurso. El siguiente código define un detector para este evento:

```
air.Localizer.localizer.addEventListener(air.Localizer.FILE_NOT_FOUND, fnfHandler);
air.Localizer.localizer.getFile("missing.html", null, "fr");
function fnfHandler(event)
{
    alert(event.fileName + ": " + event.locale);
}
```

Más temas de ayuda

Building a multilingual HTML-based application (Creación de una aplicación multilingüe basada en HTML; en inglés)